浴谷八连测 R1 题目解析

ddd

2017 年 10 月 7 日

Jelly

一个果冻，有解冻温度 c。如果当前温度小于 c，则使其升高一度需要加热 p 时间；如果当前温度恰好等于 c，使其解冻需要 q 时间；如果当前温度大于 c，使其升高一度需要加热 r 时间。现给出时间 x 和起始温度 a，问加热 x 时间后温度变成多少。

Jelly

一个果冻，有解冻温度 c。如果当前温度小于 c，则使其升高一度需要加热 p 时间；如果当前温度恰好等于 c，使其解冻需要 q 时间；如果当前温度大于 c，使其升高一度需要加热 r 时间。现给出时间 x 和起始温度 a，问加热 x 时间后温度变成多少。

- 对于 30% 的数据，$|a|, |c| \leq 200, x \leq 100$。

Jelly

一个果冻，有解冻温度 c。如果当前温度小于 c，则使其升高一度需要加热 p 时间；如果当前温度恰好等于 c，使其解冻需要 q 时间；如果当前温度大于 c，使其升高一度需要加热 r 时间。现给出时间 x 和起始温度 a，问加热 x 时间后温度变成多少。

- 对于 30% 的数据，$|a|, |c| \le 200, x \le 100$。
- 对于 60% 的数据，$|a|, |c| \le 2 \times 10^9, x \le 100$。

Jelly

一个果冻，有解冻温度 c。如果当前温度小于 c，则使其升高一度需要加热 p 时间；如果当前温度恰好等于 c，使其解冻需要 q 时间；如果当前温度大于 c，使其升高一度需要加热 r 时间。现给出时间 x 和起始温度 a，问加热 x 时间后温度变成多少。

- 对于 30% 的数据，$|a|, |c| \leq 200, x \leq 100$。
- 对于 60% 的数据，$|a|, |c| \leq 2 \times 10^9, x \leq 100$。
- 对于 100% 的数据，$|a|, |c| \leq 2 \times 10^9, 1 \leq x, p, q, r \leq 10^9$。

Solution

- 如果 $a > c$，答案是 $a + \lfloor \frac{x}{r} \rfloor$。

Solution

- 如果 $a > c$，答案是 $a + \lfloor \frac{x}{r} \rfloor$。
- 如果 $a = c$，有两种情况：

Solution

- 如果 $a > c$，答案是 $a + \lfloor \frac{x}{r} \rfloor$。
- 如果 $a = c$，有两种情况：
  - 如果 $x \leq q$，答案是 c。

Solution

- 如果 $a > c$，答案是 $a + \lfloor \frac{x}{r} \rfloor$。
- 如果 $a = c$，有两种情况：
  - 如果 $x \leq q$，答案是 c。
  - 如果 $x > q$，答案是 $c + \lfloor \frac{x-q}{r} \rfloor$。

Solution

- 如果 $a > c$，答案是 $a + \lfloor \frac{x}{r} \rfloor$。
- 如果 $a = c$，有两种情况：
  - 如果 $x \leq q$，答案是 c。
  - 如果 $x > q$，答案是 $c + \lfloor \frac{x-q}{r} \rfloor$。
- 如果 $a < c$，有三种情况：

Solution

- 如果 $a > c$，答案是 $a + \lfloor \frac{x}{r} \rfloor$。
- 如果 $a = c$，有两种情况：
  - 如果 $x \le q$，答案是 c。
  - 如果 $x > q$，答案是 $c + \lfloor \frac{x-q}{r} \rfloor$。
- 如果 $a < c$，有三种情况：
  - 如果 $x \le (c - a)p$，答案是 $a + \lfloor \frac{x}{p} \rfloor$。

Solution

- 如果 $a > c$，答案是 $a + \lfloor \frac{x}{r} \rfloor$。
- 如果 $a = c$，有两种情况：
  - 如果 $x \leq q$，答案是 c。
  - 如果 $x > q$，答案是 $c + \lfloor \frac{x-q}{r} \rfloor$。
- 如果 $a < c$，有三种情况：
  - 如果 $x \leq (c-a)p$，答案是 $a + \lfloor \frac{x}{p} \rfloor$。
  - 如果 $(c-a)p \leq x \leq (c-a)p + q$，答案是 c。

Solution

- 如果 $a > c$，答案是 $a + \lfloor \frac{x}{r} \rfloor$。
- 如果 $a = c$，有两种情况：
  - 如果 $x \leq q$，答案是 c。
  - 如果 $x > q$，答案是 $c + \lfloor \frac{x-q}{r} \rfloor$。
- 如果 $a < c$，有三种情况：
  - 如果 $x \leq (c-a)p$，答案是 $a + \lfloor \frac{x}{p} \rfloor$。
  - 如果 $(c-a)p \leq x \leq (c-a)p + q$，答案是 c。
  - 如果 $x > (c-a)p + q$，答案是 $c + \lfloor \frac{x-(c-a)p-q}{r} \rfloor$。

Solution

- 如果 $a > c$，答案是 $a + \lfloor \frac{x}{r} \rfloor$。
- 如果 $a = c$，有两种情况：
  - 如果 $x \le q$，答案是 c。
  - 如果 $x > q$，答案是 $c + \lfloor \frac{x-q}{r} \rfloor$。
- 如果 $a < c$，有三种情况：
  - 如果 $x \le (c-a)p$，答案是 $a + \lfloor \frac{x}{p} \rfloor$。
  - 如果 $(c-a)p \le x \le (c-a)p + q$，答案是 c。
  - 如果 $x > (c-a)p + q$，答案是 $c + \lfloor \frac{x-(c-a)p-q}{r} \rfloor$。
- 时间复杂度 $O(1)$。

Sequence1

定义一个序列为" 波动序列 “，当该序列满足以下两个条件之一：

条件 A：对于所有的 i，有 $a_{2i} > a_{2i-1}$ 且 $a_{2i} > a_{2i+1}$。

条件 B：对于所有的 i，有 $a_{2i} < a_{2i-1}$ 且 $a_{2i} < a_{2i+1}$。

现给出一个序列，问能否进行不大于 1 次修改，使得该序列成为波动序列。

Sequence1

定义一个序列为" 波动序列 "，当该序列满足以下两个条件之一：

条件 A：对于所有的 i，有 $a_{2i} > a_{2i-1}$ 且 $a_{2i} > a_{2i+1}$。

条件 B：对于所有的 i，有 $a_{2i} < a_{2i-1}$ 且 $a_{2i} < a_{2i+1}$。

现给出一个序列，问能否进行不大于 1 次修改，使得该序列成为波动序列。

- 对于 30% 的数据，$n \leq 10$。

Sequence1

定义一个序列为" 波动序列 "，当该序列满足以下两个条件之一：

条件 A：对于所有的 i，有 $a_{2i} > a_{2i-1}$ 且 $a_{2i} > a_{2i+1}$。

条件 B：对于所有的 i，有 $a_{2i} < a_{2i-1}$ 且 $a_{2i} < a_{2i+1}$。

现给出一个序列，问能否进行不大于 1 次修改，使得该序列成为波动序列。

- 对于 30% 的数据，$n \leq 10$。
- 对于另外 30% 的数据，$\max(|a_i|) \leq 1000$。

Sequence1

定义一个序列为" 波动序列 "，当该序列满足以下两个条件之一：

条件 A：对于所有的 i，有 $a_{2i} > a_{2i-1}$ 且 $a_{2i} > a_{2i+1}$。

条件 B：对于所有的 i，有 $a_{2i} < a_{2i-1}$ 且 $a_{2i} < a_{2i+1}$。

现给出一个序列，问能否进行不大于 1 次修改，使得该序列成为波动序列。

- 对于 30% 的数据，$n \leq 10$。
- 对于另外 30% 的数据，$\max(|a_i|) \leq 1000$。
- 对于 100% 的数据，$1 \leq n \leq 10^5, \max(|a_i|) \leq 10^9$。

Solution: 30 分

- 如果给定一个序列，可以很容易的在 $O(n)$ 时间内判断该序列是否为波动序列。

Solution: 30 分

- 如果给定一个序列，可以很容易的在 $O(n)$ 时间内判断该序列是否为波动序列。
- 首先判断该序列是否为波动序列，如果是，则直接输出"Yes"。

Solution: 30 分

- 如果给定一个序列，可以很容易的在 $O(n)$ 时间内判断该序列是否为波动序列。
- 首先判断该序列是否为波动序列，如果是，则直接输出"Yes"。
- 否则，枚举修改哪一个数。

Solution：30 分

- 如果给定一个序列，可以很容易的在 $O(n)$ 时间内判断该序列是否为波动序列。
- 首先判断该序列是否为波动序列，如果是，则直接输出"Yes"。
- 否则，枚举修改哪一个数。
- 可以发现如一个数要被修改，则将其改为 ∞ 或 $-\infty$ 一定不会比修改为别的数不优。

Solution: 30 分

- 如果给定一个序列，可以很容易的在 $O(n)$ 时间内判断该序列是否为波动序列。
- 首先判断该序列是否为波动序列，如果是，则直接输出"Yes"。
- 否则，枚举修改哪一个数。
- 可以发现如一个数要被修改，则将其改为 ∞ 或 $-\infty$ 一定不会比修改为别的数不优。
- 所以将其修改为 ∞ 或 $-\infty$ 后再次判断。

Solution: 30 分

- 如果给定一个序列，可以很容易的在 $O(n)$ 时间内判断该序列是否为波动序列。
- 首先判断该序列是否为波动序列，如果是，则直接输出"Yes"。
- 否则，枚举修改哪一个数。
- 可以发现如一个数要被修改，则将其改为 ∞ 或 $-\infty$ 一定不会比修改为别的数不优。
- 所以将其修改为 ∞ 或 $-\infty$ 后再次判断。
- 总复杂度 $O(n^2)$。

Solution: 100 分

- 由于波动序列本质上只有 2 种，所以对于每一种波动序列，求出将原序列变为这种波动序列最少需要修改几次。如果两个值的较小值不大于 1，则输出"Yes"，否则输出"No"。

Solution: 100 分

- 由于波动序列本质上只有 2 种，所以对于每一种波动序列，求出将原序列变为这种波动序列最少需要修改几次。如果两个值的较小值不大于 1，则输出"Yes"，否则输出"No"。
- 问题变为求原序列变为某种波动序列需要的最小修改次数。

Solution: 100 分

- 由于波动序列本质上只有 2 种，所以对于每一种波动序列，求出将原序列变为这种波动序列最少需要修改几次。如果两个值的较小值不大于 1，则输出”Yes“，否则输出”No“。
- 问题变为求原序列变为某种波动序列需要的最小修改次数。
- 从前向后扫，如果遇到某个元素不满足要求，则将该元素修改为 ∞ 和 $-\infty$ 中满足要求的那个，并将计数器加一。

Solution: 100 分

- 由于波动序列本质上只有 2 种，所以对于每一种波动序列，求出将原序列变为这种波动序列最少需要修改几次。如果两个值的较小值不大于 1，则输出"Yes"，否则输出"No"。
- 问题变为求原序列变为某种波动序列需要的最小修改次数。
- 从前向后扫，如果遇到某个元素不满足要求，则将该元素修改为 ∞ 和 $-\infty$ 中满足要求的那个，并将计数器加一。
- 最后计数器的值就是修改需要的最小次数。

Solution: 100 分

- 由于波动序列本质上只有 2 种，所以对于每一种波动序列，求出将原序列变为这种波动序列最少需要修改几次。如果两个值的较小值不大于 1，则输出”Yes“，否则输出”No“。
- 问题变为求原序列变为某种波动序列需要的最小修改次数。
- 从前向后扫，如果遇到某个元素不满足要求，则将该元素修改为 ∞ 和 $-\infty$ 中满足要求的那个，并将计数器加一。
- 最后计数器的值就是修改需要的最小次数。
- 总复杂度 $O(n)$。

Tree

给出一棵有根树，每条边有一个权值。现要割掉若干条边，割掉一条边的代价为这条边的权值，使得没有一条路径，可以从根节点到达叶子结点。求出最小代价。

Tree

给出一棵有根树，每条边有一个权值。现要割掉若干条边，割掉一条边的代价为这条边的权值，使得没有一条路径，可以从根节点到达叶子结点。求出最小代价。

- 对于 20% 的数据，$1 \le n \le 10$。

Tree

给出一棵有根树，每条边有一个权值。现要割掉若干条边，割掉一条边的代价为这条边的权值，使得没有一条路径，可以从根节点到达叶子结点。求出最小代价。

- 对于 20% 的数据，$1 \le n \le 10$。
- 对于 50% 的数据，$1 \le n \le 1000$。

Tree

给出一棵有根树，每条边有一个权值。现要割掉若干条边，割掉一条边的代价为这条边的权值，使得没有一条路径，可以从根节点到达叶子结点。求出最小代价。

- 对于 20% 的数据，$1 \le n \le 10$。
- 对于 50% 的数据，$1 \le n \le 1000$。
- 对于 100% 的数据，$1 \le n \le 10^5$。

Solution: 20 分

- 每条边只有割掉或不割掉两个状态。

Solution: 20 分

- 每条边只有割掉或不割掉两个状态。
- 枚举每条边割或不割，判断是否满足题意，并更新答案。

Solution: 20 分

- 每条边只有割掉或不割掉两个状态。
- 枚举每条边割或不割，判断是否满足题意，并更新答案。
- 总复杂度 $O(2^n n)$。

Solution: 50 分

- 根据最大流 -最小割定理，题目所求即为根到所有叶子结点的最大流。

Solution: 50 分

- 根据最大流 -最小割定理，题目所求即为根到所有叶子结点的最大流。
- 源点连向根，所有叶子结点连向汇点，最大流即为答案。

Solution: 50 分

- 根据最大流 -最小割定理，题目所求即为根到所有叶子结点的最大流。
- 源点连向根，所有叶子结点连向汇点，最大流即为答案。
- 如果使用比较主流的最大流算法，总复杂度 $O(n^3)$。

Solution: 100 分

- 设 dp_i 表示：割断 i 和 i 的子树中所有叶子节点所需的最小代价。

Solution: 100 分

- 设 dp_i 表示：割断 i 和 i 的子树中所有叶子节点所需的最小代价。
- 对于叶子节点，$dp_i = \infty$。

Solution: 100 分

- 设 dp_i 表示：割断 i 和 i 的子树中所有叶子节点所需的最小代价。
- 对于叶子节点，$dp_i = \infty$。
- 否则，$dp_i = \sum_j \min(dp_j, val_{i,j})$，($j$ 是 i 的儿子节点)。

Solution: 100 分

- 设 dp_i 表示：割断 i 和 i 的子树中所有叶子节点所需的最小代价。
- 对于叶子节点，$dp_i = \infty$。
- 否则，$dp_i = \sum_j \min(dp_j, val_{i,j})$，($j$ 是 i 的儿子节点)。
- 答案即为 dp_{root}。

Solution: 100 分

- 设 dp_i 表示：割断 i 和 i 的子树中所有叶子节点所需的最小代价。
- 对于叶子节点，$dp_i = \infty$。
- 否则，$dp_i = \sum_j \min(dp_j, val_{i,j})$，($j$ 是 i 的儿子节点)。
- 答案即为 dp_{root}。
- 总复杂度 $O(n)$。

Factorial

求 $n!$ 在 k 进制下后缀 0 的个数。

Factorial

求 $n!$ 在 k 进制下后缀 0 的个数。

- 对于 20% 的数据，$1 \le n \le 10^6, k = 10$。

Factorial

求 $n!$ 在 k 进制下后缀 0 的个数。

- 对于 20% 的数据，$1 \le n \le 10^6, k = 10$。
- 对于另外 20% 的数据，$1 \le n \le 20, 2 \le k \le 36$。

Factorial

求 $n!$ 在 k 进制下后缀 0 的个数。

- 对于 20% 的数据，$1 \le n \le 10^6, k = 10$。
- 对于另外 20% 的数据，$1 \le n \le 20, 2 \le k \le 36$。
- 对于 60% 的数据，$1 \le n \le 10^{15}, 2 \le k \le 10^{12}$。

Factorial

求 $n!$ 在 k 进制下后缀 0 的个数。

- 对于 20% 的数据，$1 \leq n \leq 10^6, k = 10$。
- 对于另外 20% 的数据，$1 \leq n \leq 20, 2 \leq k \leq 36$。
- 对于 60% 的数据，$1 \leq n \leq 10^{15}, 2 \leq k \leq 10^{12}$。
- 对于 100% 的数据，$1 \leq n \leq 10^{18}, 2 \leq k \leq 10^{16}$。

Solution: 20 分

- 将 $n!$ 表示成 $x \times 2^y 5^z$ 的形式，其中 $x \mod 2 \neq 0, x \mod 5 \neq 0$，则答案为 $\min(y, z)$。

Solution: 20 分

- 将 $n!$ 表示成 $x \times 2^y 5^z$ 的形式，其中 $x \mod 2 \neq 0, x \mod 5 \neq 0$，则答案为 $\min(y, z)$。
- 所以只需要统计 $n!$ 的质因数分解的结果中，2 和 5 的次数即可。

Solution: 20 分

- 将 $n!$ 表示成 $x \times 2^y 5^z$ 的形式，其中 $x \mod 2 \neq 0, x \mod 5 \neq 0$，则答案为 $\min(y, z)$。
- 所以只需要统计 $n!$ 的质因数分解的结果中，2 和 5 的次数即可。
- 由于数的乘法对应的是指数的加法，所以求出 $[1, n]$ 中所有数的质因数分解的结果中，2 和 5 的次数，再作和即可。

Solution: 20 分

- 将 $n!$ 表示成 $x \times 2^y 5^z$ 的形式，其中 $x \mod 2 \neq 0, x \mod 5 \neq 0$，则答案为 $\min(y, z)$。
- 所以只需要统计 $n!$ 的质因数分解的结果中，2 和 5 的次数即可。
- 由于数的乘法对应的是指数的加法，所以求出 $[1, n]$ 中所有数的质因数分解的结果中，2 和 5 的次数，再作和即可。
- 求一个数 x 中有多少个 p，可以在 $O(\log x)$ 时间内完成。

Solution: 20 分

- 将 $n!$ 表示成 $x \times 2^y 5^z$ 的形式，其中 $x \mod 2 \neq 0, x \mod 5 \neq 0$，则答案为 $\min(y, z)$。
- 所以只需要统计 $n!$ 的质因数分解的结果中，2 和 5 的次数即可。
- 由于数的乘法对应的是指数的加法，所以求出 $[1, n]$ 中所有数的质因数分解的结果中，2 和 5 的次数，再作和即可。
- 求一个数 x 中有多少个 p，可以在 $O(\log x)$ 时间内完成。
- 总时间复杂度 $O(n \log n)$。

Solution: 40 分

- 手写高精度，模拟 k 进制下的乘法。可以拿到第二个 20% 的分数。

Solution: 60 分

- 设 k 的质因数分解的结果为 $p_1^{a_1} p_2^{a_2} \dots p_t^{a_t}$。

Solution: 60 分

- 设 k 的质因数分解的结果为 $p_1^{a_1}p_2^{a_2}...p_t^{a_t}$。
- 将 $n!$ 表示为 $x \times p_1^{b_1}p_2^{b_2}...p_t^{b_t}$，满足对于任意的 $i \in [1, t]$，$x \bmod p_i \neq 0$。

Solution: 60 分

- 设 k 的质因数分解的结果为 $p_1^{a_1} p_2^{a_2} \ldots p_t^{a_t}$。
- 将 $n!$ 表示为 $x \times p_1^{b_1} p_2^{b_2} \ldots p_t^{b_t}$，满足对于任意的 $i \in [1, t]$，$x \bmod p_i \neq 0$。
- 则答案即为 $\min\{\lfloor \frac{b_i}{a_i} \rfloor\}, i \in [1, t]$。

Solution: 60 分

- 设 k 的质因数分解的结果为 $p_1^{a_1}p_2^{a_2}\dots p_t^{a_t}$。
- 将 $n!$ 表示为 $x \times p_1^{b_1}p_2^{b_2}\dots p_t^{b_t}$，满足对于任意的 $i \in [1,t]$，$x \bmod p_i \neq 0$。
- 则答案即为 $\min\{\lfloor\frac{b_i}{a_i}\rfloor\}, i \in [1,t]$。
- 如何快速统计 $n!$ 中有多少个 p？

Solution: 60 分

- 设 k 的质因数分解的结果为 $p_1^{a_1}p_2^{a_2}...p_t^{a_t}$。
- 将 $n!$ 表示为 $x \times p_1^{b_1}p_2^{b_2}...p_t^{b_t}$，满足对于任意的 $i \in [1, t]$，$x \bmod p_i \neq 0$。
- 则答案即为 $\min\{\lfloor\frac{b_i}{a_i}\rfloor\}, i \in [1, t]$。
- 如何快速统计 $n!$ 中有多少个 p？
- $ans = \sum_{i=1}^{\infty} \frac{n}{p^i}$。

Solution: 60 分

- 设 k 的质因数分解的结果为 $p_1^{a_1}p_2^{a_2}\dots p_t^{a_t}$。
- 将 $n!$ 表示为 $x \times p_1^{b_1}p_2^{b_2}\dots p_t^{b_t}$，满足对于任意的 $i \in [1,t]$，$x \bmod p_i \neq 0$。
- 则答案即为 $\min\{\lfloor \frac{b_i}{a_i} \rfloor\}, i \in [1,t]$。
- 如何快速统计 $n!$ 中有多少个 p？
- $ans = \sum_{i=1}^{\infty} \frac{n}{p^i}$。
- 一般的质因数分解的复杂度为 $O(\sqrt{k})$，故总复杂度 $O(\sqrt{k} + \log n)$。

Solution: 100 分

- 使用 Pollard-Rho 算法将质因数分解的复杂度优化为 $O(k^{\frac{1}{3}})$

Solution: 100 分

- 使用 Pollard-Rho 算法将质因数分解的复杂度优化为 $O(k^{\frac{1}{3}})$
- 总复杂度 $O(k^{\frac{1}{3}} + \log n)$。

Sequence2

给出三个长度为 n 的序列，从这三个序列中按顺序选取最多个元素，组成一个子序列，满足：

如果子序列的一个元素来自第一个序列，则该元素不小于子序列中的前一个元素。

如果子序列的一个元素来自第二个序列，则该元素不大于子序列中的前一个元素。

如果子序列的一个元素来自第三个序列，则该元素和前一个元素的增减性，与前一个元素和前两个元素的增减性相同。

Sequence2

给出三个长度为 n 的序列，从这三个序列中按顺序选取最多个元素，组成一个子序列，满足：

如果子序列的一个元素来自第一个序列，则该元素不小于子序列中的前一个元素。

如果子序列的一个元素来自第二个序列，则该元素不大于子序列中的前一个元素。

如果子序列的一个元素来自第三个序列，则该元素和前一个元素的增减性，与前一个元素和前两个元素的增减性相同。

- 对于 20% 的数据，$1 \leq n \leq 10, \max(|a_i|) \leq 1000$。

Sequence2

给出三个长度为 n 的序列，从这三个序列中按顺序选取最多个元素，组成一个子序列，满足：

如果子序列的一个元素来自第一个序列，则该元素不小于子序列中的前一个元素。

如果子序列的一个元素来自第二个序列，则该元素不大于子序列中的前一个元素。

如果子序列的一个元素来自第三个序列，则该元素和前一个元素的增减性，与前一个元素和前两个元素的增减性相同。

- 对于 20% 的数据，$1 \le n \le 10, \max(|a_i|) \le 1000$。
- 对于 60% 的数据，$1 \le n \le 1000, \max(|a_i|) \le 1000$。

Sequence2

给出三个长度为 n 的序列，从这三个序列中按顺序选取最多个元素，组成一个子序列，满足：

如果子序列的一个元素来自第一个序列，则该元素不小于子序列中的前一个元素。

如果子序列的一个元素来自第二个序列，则该元素不大于子序列中的前一个元素。

如果子序列的一个元素来自第三个序列，则该元素和前一个元素的增减性，与前一个元素和前两个元素的增减性相同。

- 对于 20% 的数据，$1 \le n \le 10, \max(|a_i|) \le 1000$。
- 对于 60% 的数据，$1 \le n \le 1000, \max(|a_i|) \le 1000$。
- 对于 100% 的数据，$1 \le n \le 10^5, \max(|a_i|) \le 10^9$。

Solution: 20 分

- 每个位置有四种状态：不选、选第一个序列，选第二个序列，选第三个序列。

Solution: 20 分

- 每个位置有四种状态：不选、选第一个序列，选第二个序列，选第三个序列。
- 暴力枚举每种状态，判断是否满足题意，同时更新答案。

Solution: 20 分

- 每个位置有四种状态：不选、选第一个序列，选第二个序列，选第三个序列。
- 暴力枚举每种状态，判断是否满足题意，同时更新答案。
- 时间复杂度 $O(4^n n)$。

Solution: 60 分

- 设 $dp_{i,j}$ 表示：第 i 个位置选择第 j 个序列的数，前 i 个位置所能构成的最长子序列的长度。特别的，$dp_{i,2}$ 表示第 i 个位置选择第三个序列的数，且大于等于前一个数；$dp_{i,3}$ 表示第 i 个位置选择第三个序列的数，且小于等于前一个数。

Solution: 60 分

- 设 $dp_{i,j}$ 表示：第 i 个位置选择第 j 个序列的数，前 i 个位置所能构成的最长子序列的长度。特别的，$dp_{i,2}$ 表示第 i 个位置选择第三个序列的数，且大于等于前一个数；$dp_{i,3}$ 表示第 i 个位置选择第三个序列的数，且小于等于前一个数。
- $dp_{i,0} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, a_{j,k} \le a_{i,0}) + 1$
  $dp_{i,1} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, a_{j,k} \ge a_{i,1}) + 1$
  $dp_{i,2} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, k \ne 3, a_{j,k} \le a_{i,2}) + 1$
  $dp_{i,3} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, k \ne 2, a_{j,k} \ge a_{i,2}) + 1$

Solution：60 分

- 设 $dp_{i,j}$ 表示：第 i 个位置选择第 j 个序列的数，前 i 个位置所能构成的最长子序列的长度。特别的，$dp_{i,2}$ 表示第 i 个位置选择第三个序列的数，且大于等于前一个数；$dp_{i,3}$ 表示第 i 个位置选择第三个序列的数，且小于等于前一个数。
- $dp_{i,0} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, a_{j,k} \le a_{i,0}) + 1$
  $dp_{i,1} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, a_{j,k} \ge a_{i,1}) + 1$
  $dp_{i,2} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, k \ne 3, a_{j,k} \le a_{i,2}) + 1$
  $dp_{i,3} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, k \ne 2, a_{j,k} \ge a_{i,2}) + 1$
- 边界条件：$dp_{0,j} = 0$。

Solution: 60 分

- 设 $dp_{i,j}$ 表示：第 i 个位置选择第 j 个序列的数，前 i 个位置所能构成的最长子序列的长度。特别的，$dp_{i,2}$ 表示第 i 个位置选择第三个序列的数，且大于等于前一个数；$dp_{i,3}$ 表示第 i 个位置选择第三个序列的数，且小于等于前一个数。
- $dp_{i,0} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, a_{j,k} \le a_{i,0}) + 1$
  $dp_{i,1} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, a_{j,k} \ge a_{i,1}) + 1$
  $dp_{i,2} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, k \ne 3, a_{j,k} \le a_{i,2}) + 1$
  $dp_{i,3} = \max(dp_{j,k}), (0 \le j < i, 0 \le k < 4, k \ne 2, a_{j,k} \ge a_{i,2}) + 1$
- 边界条件：$dp_{0,j} = 0$。
- 状态数 $O(n)$，转移复杂度 $O(n)$，总复杂度 $O(n^2)$。

Solution: 100 分

- 该转移和 LIS 的转移类似，可以使用数据结构进行优化。

Solution: 100 分

- 该转移和 LIS 的转移类似，可以使用数据结构进行优化。
- 以 LIS 为例，建立一个线段树，初始时所有位置都为 0。

Solution: 100 分

- 该转移和 LIS 的转移类似，可以使用数据结构进行优化。
- 以 LIS 为例，建立一个线段树，初始时所有位置都为 0。
- 每次求 dp 值时，查询值介于 $[-\infty, a_i]$ 中的 dp 值的最大值。

Solution: 100 分

- 该转移和 LIS 的转移类似，可以使用数据结构进行优化。
- 以 LIS 为例，建立一个线段树，初始时所有位置都为 0。
- 每次求 dp 值时，查询值介于 $[-\infty, a_i]$ 中的 dp 值的最大值。
- 再将 i 位置的 dp 值插入到线段树 a_i 处。

Solution: 100 分

- 该转移和 LIS 的转移类似，可以使用数据结构进行优化。
- 以 LIS 为例，建立一个线段树，初始时所有位置都为 0。
- 每次求 dp 值时，查询值介于 $[-\infty, a_i]$ 中的 dp 值的最大值。
- 再将 i 位置的 dp 值插入到线段树 a_i 处。
- 转移的复杂度优化为 $O(\log n)$，故总复杂度 $O(n \log n)$。

Knight

给出一个 $n \times n$ 的国际象棋棋盘，上面有若干个黑子，和一个白骑士。黑子不动，问白子最少跳多少次可以吃掉黑子的国王。要求中途白子不能被黑子吃掉。

Knight

给出一个 $n \times n$ 的国际象棋棋盘，上面有若干个黑子，和一个白骑士。黑子不动，问白子最少跳多少次可以吃掉黑子的国王。要求中途白子不能被黑子吃掉。

- 对于 20% 的数据，$1 \leq n \leq 8$，黑子只有国王。

Knight

给出一个 $n \times n$ 的国际象棋棋盘，上面有若干个黑子，和一个白骑士。黑子不动，问白子最少跳多少次可以吃掉黑子的国王。要求中途白子不能被黑子吃掉。

- 对于 20% 的数据，$1 \le n \le 8$，黑子只有国王。
- 对于 30% 的数据，$1 \le n \le 8$，黑子只有国王和骑士。

Knight

给出一个 $n \times n$ 的国际象棋棋盘，上面有若干个黑子，和一个白骑士。黑子不动，问白子最少跳多少次可以吃掉黑子的国王。要求中途白子不能被黑子吃掉。

- 对于 20% 的数据，$1 \leq n \leq 8$，黑子只有国王。
- 对于 30% 的数据，$1 \leq n \leq 8$，黑子只有国王和骑士。
- 对于 60% 的数据，$1 \leq n \leq 50$，黑子只有国王、骑士和皇后。

Knight

给出一个 $n \times n$ 的国际象棋棋盘，上面有若干个黑子，和一个白骑士。黑子不动，问白子最少跳多少次可以吃掉黑子的国王。要求中途白子不能被黑子吃掉。

- 对于 20% 的数据，$1 \leq n \leq 8$，黑子只有国王。
- 对于 30% 的数据，$1 \leq n \leq 8$，黑子只有国王和骑士。
- 对于 60% 的数据，$1 \leq n \leq 50$，黑子只有国王、骑士和皇后。
- 对于 100% 的数据，$1 \leq n \leq 50$，黑子可以有全套的国际象棋棋子。

Solution: 20 分

- 黑子只有国王，白子不存在先吃掉某个黑子再吃掉国王的情况。

Solution: 20 分

- 黑子只有国王，白子不存在先吃掉某个黑子再吃掉国王的情况。
- 从起点开始 BFS，找最短路径即可。

Solution: 20 分

- 黑子只有国王，白子不存在先吃掉某个黑子再吃掉国王的情况。
- 从起点开始 BFS，找最短路径即可。
- 时间复杂度 $O(n^2)$。

Solution: 30 分

- 注意到黑骑士不可能被白骑士吃掉，因为他们的攻击范围是一样的。

Solution: 30 分

- 注意到黑骑士不可能被白骑士吃掉，因为他们的攻击范围是一样的。
- 所以和上一种情况一样，直接 BFS 即可。

Solution: 30 分

- 注意到黑骑士不可能被白骑士吃掉，因为他们的攻击范围是一样的。
- 所以和上一种情况一样，直接 BFS 即可。
- 时间复杂度 $O(n^2)$。

Solution: 80 分

- 设 $dp_{i,j,k}$ 表示：白骑士到达 (i,j)，此时黑子的存活情况为 k，所需要的最短路程。

Solution: 80 分

- 设 $dp_{i,j,k}$ 表示：白骑士到达 (i,j)，此时黑子的存活情况为 k，所需要的最短路程。
- 转移的时候，只需要判断 k 中为 1 的棋子的攻击范围。

Solution: 80 分

- 设 $dp_{i,j,k}$ 表示：白骑士到达 (i,j)，此时黑子的存活情况为 k，所需要的最短路程。
- 转移的时候，只需要判断 k 中为 1 的棋子的攻击范围。
- 如果转移到一个原本有黑子的地方，更新 k。

Solution: 80 分

- 设 $dp_{i,j,k}$ 表示：白骑士到达 (i,j)，此时黑子的存活情况为 k，所需要的最短路程。
- 转移的时候，只需要判断 k 中为 1 的棋子的攻击范围。
- 如果转移到一个原本有黑子的地方，更新 k。
- 时间复杂度 $O(2^{16}n^2)$。

Solution: 100 分

- 由于黑骑士不可能被吃掉，所以不需要记录其状态。

Solution：100 分

- 由于黑骑士不可能被吃掉，所以不需要记录其状态。
- 时间复杂度 $O(2^{14}n^2)$。

Thank you!