题目大意

我们定义一个长为 n 的序列 $A =< a_1, a_2, \ldots, a_n >$ 是一个 rainbow 序列当且仅当

$$n \leq 2$$

或者

$$\forall i = 2, 3, \ldots, n-1,\ a_i - a_{i-1} > a_{i+1} - a_i$$

现在有一个长度为 n 的序列 $R =< r_1, r_2, \ldots, r_n >$，求出 R 的所有满足是 rainbow 的子序列（即不必在 R 中连续）长度的最大值。

题解

我们定义 $\mathrm{dp}_{i,j}\ (i < j)$ 为子序列中最后两个元素为 R_i, R_j 时最长的 rainbow 子序列的长度。则我们有转移方程

$$\mathrm{dp}_{i,j} = \max_{R_k - R_i > R_j - R_i} dp_{k,i} + 1$$

题解

我们定义 $\mathrm{dp}_{i,j}$ $(i < j)$ 为子序列中最后两个元素为 R_i, R_j 时最长的 rainbow 子序列的长度。则我们有转移方程

$$\mathrm{dp}_{i,j} = \max_{R_k - R_i > R_j - R_i} dp_{k,i} + 1$$

直接转移的复杂度时 $\mathcal{O}(n^3)$ 的，使用单调栈 + 二分可以将复杂度降为 $\mathcal{O}(n^2 \log n)$。

题解

我们定义 $\text{dp}_{i,j}$ $(i < j)$ 为子序列中最后两个元素为 R_i, R_j 时最长的 rainbow 子序列的长度。则我们有转移方程

$$\text{dp}_{i,j} = \max_{R_k - R_i > R_j - R_i} dp_{k,i} + 1$$

直接转移的复杂度时 $\mathcal{O}(n^3)$ 的，使用单调栈 + 二分可以将复杂度降为 $\mathcal{O}(n^2 \log n)$。
是否还有更优的做法？

题解

$$\mathrm{dp}_{i,j} = \max_{R_k - R_i > R_j - R_i} \mathrm{dp}_{k,i} + 1$$

不妨考虑改变 j 和 k 的枚举顺序。如果 j 按照 R_j 从大到小的方式枚举，可以发现此时满足条件的 k 的集合就会变得只增不减

题解

$$\mathrm{dp}_{i,j} = \max_{R_k - R_i > R_j - R_i} \mathrm{dp}_{k,i} + 1$$

不妨考虑改变 j 和 k 的枚举顺序。如果 j 按照 R_j 从大到小的方式枚举，可以发现此时满足条件的 k 的集合就会变得只增不减，此时再改变 k 为 R_k 从小到大的顺序，就可以维护出这个集合中 $\mathrm{dp}_{k,i}$ 的最大值。

题解

$$\mathrm{dp}_{i,j} = \max_{R_k - R_i > R_j - R_i} \mathrm{dp}_{k,i} + 1$$

不妨考虑改变 j 和 k 的枚举顺序。如果 j 按照 R_j 从大到小的方式枚举，可以发现此时满足条件的 k 的集合就会变得只增不减，此时再改变 k 为 R_k 从小到大的顺序，就可以维护出这个集合中 $\mathrm{dp}_{k,i}$ 的最大值。

实现的方式有很多，比方说求出一个数组 $\{p_i\}$ 使得 $\forall i = 1, 2, \ldots, n-1,\ R_{p_i} \le R_{p_{i+1}}$。枚举 j 和 k 时便直接从 $\{p_i\}$ 枚举，如果遇到 $j < i$ 或者 $k > i$ 的情况直接 `continue` 即可。