**贪心 + 二分查找**

考虑一个简单的贪心，如果我们要使上升子序列尽可能的长，则我们需要让序列上升得尽可能慢，因此我们希望每次在上升子序列最后加上的那个数尽可能的小。

基于上面的贪心思路，我们**维护一个数组 d[i] ，表示长度为 i 的最长上升子序列的末尾元素的最小值**，用 len 记录目前最长上升子序列的长度，起始时 len 为 1，d[1]=nums[0]。

同时我们可以注意到 d[i] 是关于 i 单调递增的。因为如果 d[j]≥d[i] 且 j<i，我们考虑从长度为 i 的最长上升子序列的末尾删除 i−j 个元素，那么这个序列长度变为 j ，且第 j 个元素 x（末尾元素）必然小于 d[i]，也就小于 d[j]。那么我们就找到了一个长度为 j 的最长上升子序列，并且末尾元素比 d[j] 小，从而产生了矛盾。因此数组 d 的单调性得证。

我们依次遍历数组 nums 中的每个元素，并更新数组 d 和 len 的值。如果 nums[i]>d[len] 则更新 len=len+1，否则在 d[1…len]中找满足 d[i−1]<nums[j]<d[i] 的下标 i，并更新 d[i]=nums[j]。

根据 d 数组的单调性，我们可以使用二分查找寻找下标 i，优化时间复杂度。

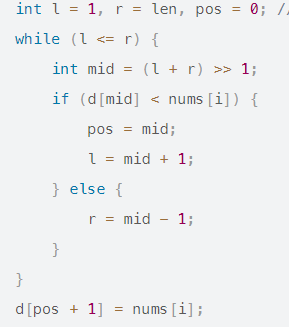
最后整个算法流程为：

设当前已求出的最长上升子序列的长度为 len（初始时为 1），从前往后遍历数组 nums，在遍历到 nums[i] 时：

如果 nums[i]>d[len] ，则直接加入到 d 数组末尾，并更新 len=len+1；

否则，在 d 数组中二分查找，找到第一个比 nums[i] 小的数 d[k] ，并更新 d[k+1]=nums[i]。

时间复杂度：*O*(*n*log*n*)



**二分的关键：l=mid+1,r=mid-1;确保mid不会重复选择l和r**

mid能访问到l和r这两个边界

二维的最长递增子序列问题：[俄罗斯套娃信封问题](https://leetcode.cn/problems/russian-doll-envelopes/solutions/633231/e-luo-si-tao-wa-xin-feng-wen-ti-by-leetc-wj68)