最长递增子序列（可不连续） LIS

不妨设待处理数组为 double a[n];
记另一数组为 dp[n] 类型为 int*，表示以第i个结尾的 LIS 长.
那么初始时有 dp[0] = 1 （自身），其余为 0.
而 dp[i] 与 dp[i+1] 的关系为 { dp[i]+1 （a[i+1] > a[i]）（1234步）
dp[j]+1 于是从i从前找第一个小于a[i+1]者.
1 （a[i+1]比前面都小）（2341）

二得方程. **$dp[i] = \max(1, \max_{0\le j\le i-1}\{dp[j]+1 \mid a[j]<a[i]\})$**

而 LIS = dp[i] 整个数组最大值.

伪代码：

```
int a[] = {3,2,3,0,1,0,4,1,4,8}; (待处理元素)
int n = 10; (问题规模)
int LIS = 1; (答案)
int dp[n]; (动态规划数组)

int LISinOn^2(int ....){
    int i,j;
    for i=0 i++ -> i<n {
        dp[i]=1
        for j=i-1 j-- -> j>0 {
            if a[j]<a[i] {
                dp[i] = max(dp[i], dp[j]+1);
            }
        }
    }
}
Return LIS.
```

找到上一个 能小的 a[i]，其+1
二重循环，复杂度为 $O(n^2)$

要使复杂度降至 O(nlogn)

第二个循环的功能是找到一个使 a[j] < a[i] 且更新其 dp 值.
所以希望将搜索复杂度降至 O(logn)

所以建立一个 一个长度 ⟼ 该长度子列的末尾的最小值. 的 map.
称 least[n] (该数组不长于答案即LIS值)

对每个 a[i]

从 least 中进行二分查找, 找到 least ∩ [a[i], +∞)
的最大值的 index.
而 dp[i] 与 index+1 取最大值.
而 least[index+1] 与 a[i] 取小值
实现 least 的更新.

如 a 3 2 3 0 1 0 4 1 4 8.
dp 1 1 2 1 2 1 3 0 0 0
least ??? 3 4 6 ??? ??? .....

当处理到 a[7] 时.
least 中表示 长度为i的从此位前的最末子列末的末.
尾数字的最小值为 a[least[i]] 处的数.
如 要接着"4"往下做长度为 3+1 的 LIS, 那
起码要大于 a[least[3]], 即大于"4"

∴把搜索从 O(n²) 优化至了二分查找的 O(nlogn)