时间复杂度为 $O(n^2)$

原序列存放在 nums 中，长度为 size

length[i] 表示以 nums[i] 结尾的最长严格单调递增子序列长度

path[j] 表示子序列中 nums[j] 前一个数的索引，不存在为 -1

```
for i=0 to size-1:
    length[i]=1; path[i]=-1;
    for j=0 to i-1:
        if nums[j] < nums[i] and length[j]+1 > length[i]:
            length[i] = length[j]+1
            path[i] = j
```

结束循环后，length 数组最大元的值即为 LIS 的长度，比最大元索引为 max

```
k = max
存储 nums[k]
while k ≠ -1:
    存储 nums[path[k]]
    k = path[k]
```

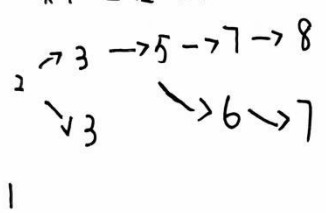
最后翻转得到的序列即为 LIS。多个 LIS 的情况也可通过这种方式得到。

举个例：

索引：	⓪	①	②	③	④	⑤	⑥	⑦	⑧
nums：	2	3	1	5	7	8	3	6	7
length：	1	2	1	3	4	5	2	4	5
path：	-1	0	-1	1	3	4	0	3	7

存在2个长度为5的LIS：2 3 5 7 8 和 2 3 5 6 7

实际上在做



时间复杂度为O(nlogn)

原序列存放在nums中，长度为size

维护一个数组 cell，cell[i]表示长度为i的LIS末尾元素的最小值

```
k = 1
cell[1] = nums[0]

for i = 1 to size-1:
    if nums[i] > cell[k]   长度为k的LIS严格单增，比末尾元素大则比该序列的所有元素大，直接连接在末尾
        ++k
        cell[k] = nums[i]
    else   二分查找，替换比nums[i]大的所有元素中最小的那个（由cell[i]的定义可知这是合理的）
        left = 1, right = k, pos = 0
        while left < right:
            mid = [(left+right)/2]
            if cell[mid] < nums[i]
                left = mid+1, pos = mid;
            else
                right = mid
        cell[pos+1] = nums[i]
```

循环结束后，k的值即为LIS的长度。

举例，nums 10 9 2 5 3 7 21 18

cell 2 3 7 18

↑

k=4，LIS长度为4