3230102918 姚杭帝

最长严格递增子序列

```
int n = nums.length    // 先假设nums为非空数组
  res = 1
if n == 1    return 1
if n == 0    return 0    //处理特殊情况
int dp[n]                // dp表示以第i个数为结尾
                         和最长严格递增子序列长度
for i = 0~n   dp[i] = 1

for i = 1~n
    for j = 1~i

        if nums[i] > nums[j]    // nums[i]比nums[j]大可更新
        dp[i] = max(dp[i], dp[j]+1)
        res = max(dp[i], res)    //res即为答案
cout << res.
```

示例 nums[0,1,0,3,2,3]

dp[1,1,1,1,1,1] 初始

① i=1
dp[i] = dp[0]+1 = 2 res=2

② i=2
dp[i] 不变

③ i=3
dp[i] = max(dp[0]+1, dp[1]+1, dp[2]+1)
= 3
res=3

④ i=4
dp[i] = max(dp[0]+1, dp[1]+1, dp[2]+1)
=3
res=3

⑤ i=5
dp[i] = max(dp[0]+1, dp[1]+1, dp[2]+1,
dp[4]+1 = 4
res=4

⑥ res=4

此算法为 $O(n^2)$

[9,3,5]

改进：我们发现最后的值越小越好

因此我们可以新建一个 d[len]数组

表示：长为len的严格递增子序列末尾最小为多少

```
len=1
  d[1] = nums[0]
  for i = 1~n
     if(nums[i] > d[len])
          len++
          d[len] = nums[i]
   else
       l=1, r = len, pos=0   //采用二分法降低时间复杂度)
       while l <= r
          mid = (l+r)/2
          if d[mid] < nums[i]
              pos=mid
              l=mid+1
          else
              r=mid-1
       d[pos+1] = nums[i] //找到d中第一个小于nums[i]的数，将后一个设为nums[i]
  return len
```

示例：10 9 2 5 3 7 101 18 19

① d[1] = 10

② i=1
d[1] = 9

③ i=2
d[1] = 2

④ i=3
d[2] = 5   len=2

⑤ i=4
d[2] = 3

⑥ i=5
d[3] = 7   len=3

⑦ i=6
d[4] = 101   len=4

⑧ i=7
d[4] = 18   len=4

⑨ i=8
d[5] = 19   len=5

2, 3, 7, 18, 19

此算法复杂度为 $O(n\log n)$