LIS 问题

给定序列 $A=a_1,\dots,a_n$，找最长递增子序列 LIS

找到的LIS的最后一个元素是LIS中最大的

定义一维数组 L[n+1]，L[i] 是以 a_i 结尾的递增子序列长度，初始化为1

```
def L(A,i,j):        // i<j
    if ai<aj:
        L[j] = max(L[i]+1, L[j]);

for j in range(1,n):
    for i in range(0,j):
        L(A,i,j);
```

则能得到正确的 L，找到最大的 L[i]

例：A=[7 1 2 3 4 9 7 8]　　n=8

初始化 L=[1,1,1,1,1,1,1,1]

j	a_j	L[0]	L[1]	L[2]	L[3]	L[4]	L[5]	L[6]	L[7]
0	7	1							
1	1		①						
2	2			②					
3	3				③				
4	4					④			
5	9						5		
6	7							⑤	
7	8								⑥

j=5时，L[5]=1

$a_0<a_5$　L[5]=max(L[0]+1, L[5])=2

$a_1<a_5$　L[5]=max[L[1]+1, L[5])=2

$a_2<a_5$　L[5]=max[L[2]+1, L[5])=3

$a_3<a_5$　L[5]=max[L[3]+1, L[5])=4

$a_4<a_5$　L[5]=max[L[4]+1, L[5])=5

if L[i]=L[i-1]+1，则 a_{i-1} 在 LIS 中

时间复杂度为 $O(n^2)$

O(nlogn)算法

给定序列 $A=a_1,a_2,\dots,a_n$

定义数组 tail，tail[i]表示长度为i+1的所有递增子序列中最小的末尾元素，若不存在长度为i+1的递增子序列，tail[i]=+∞，path记录位置

初始化 tail[0]=A[0]

断言 tail为单增序列，否则 当 tail[i+1] ≤ tail[i] 时

设长度为i+1的递增子序列为 $c_1,\dots,c_i$, tail[i]

设长度为i+2的递增子序列为 $c_1',\dots,c_i',c_{i+1}'$, tail[i+1]

则 $c_{i+1}' < tail[i+1] \le tail[i]$

$c_1',\dots,c_{i+1}'$ 的长度为i+1，则 $c_{i+1}' \ge tail[i]$，矛盾

故对于tail中的元素，可使用二分查找

```
for i in range(1,n):
    idx = tail.find(a_i) ⇒ 返回tail中第一个大于等于a_i的元素位置
    if idx == tail.size():    // a_i > tail.all
        tail.append(a_i);
    else:                     // tail[idx] ≥ a_i，例：1,2,3,5,4
                              //   tail=[1,2,3,5] → tail=[1,2,3,4]
        tail[idx] = a_i;
```

用path记录路径

LIS的长度即为 tail.size()

例 A=[10,9,2,5,3,7,1,18]

path=[-1,-1,-1,2,2,3,-1,7]

tail

10

9

2

2 5

2 3

2 3 7

1 3 7

1 3 7 18

tail序列长度+1，path记录前一个元素

tail元素替换，path记录原来元素的path

LIS=[2,3,7,18]