**一、数组**

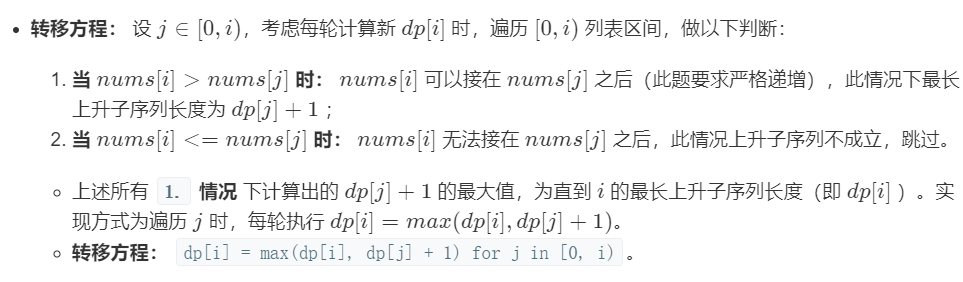
**二、Dp**

1. **最长上升子序列**

Given an unsorted array of integers, find the length of longest increasing subsequence.

①一般思路：O（N^2）

初始化：dp[i]=1;



**dp思路：从已只上升子序列里找到最长的,然后判断是否接上去，转移方程得到的为dp[i]为以nums[i]结尾的最长上升的子序列长度，从中dp找到最长的；**

**转移方程：dp[i]=Math.Max(dp[i],dp[j]+1)**

**核心算法：**

**for(int i=0;i<length;i++)**

**{**

**dp[i]=1;**

**for(int j=0;j<i;j++)**

**{**

**if(nums[j]<nums[i])**

**{**

**dp[i]=Math.Max(dp[i],dp[j]+1);**

**}**

**}**

**if(max<dp[i])**

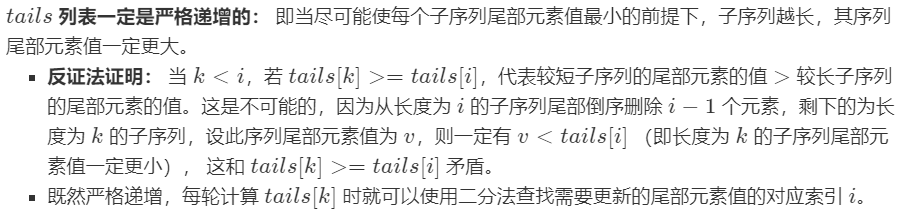
**max=dp[i];**

**}**

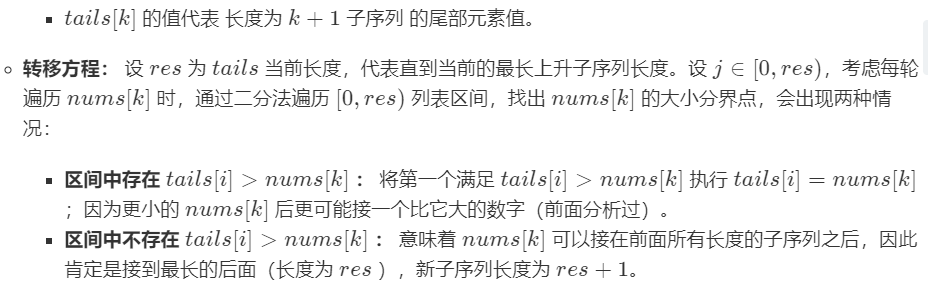
②进阶思路 O(N\*LogN)

设tails数组来记录tails[k]长度为k+1的**子序列的最小尾部值**

**证明：tails数组一定严格递增：**



**转移方程思路：**替换tails数组中的各个长度的尾部数（**二分查找**），实现最小尾部数，来达到让tails数组长度不断增加的目的，从而找到最长子序列即tails数组的长度



**核心算法：**

for(int i=1;i<length;i++)

{

if(tails[res]<nums[i])

tails[++res]=nums[i];**//大于最大直接放在后面**

else

{

int j=0,k=res;

int mid=(k+j)/2;

while(j<k)

{

if(tails[mid]<nums[i])

{

j=mid+1;

}

else if(tails[mid]>nums[i])

{

k=mid-1;

}

else

break;

mid=(j+k)/2;

}

**//找到mid满足nums[i]在区间中间**

if(tails[mid]<nums[i])

tails[mid+1]=nums[i];

else

tails[mid]=nums[i];

}

}