最长不下降子序列(LIS:Longest Increasing Subsequence)

//用句通俗的话说，我讲的很通俗易懂~~

问题描述：给出n个数，求出其最长不下降子序列的长度，比如n=5，5个数是{4,6,5,7,3}；其最长下降子序列就是{4,6,7}，长度为3。

一、简单的O（n^2)的算法

         很容易想到用动态规划做。设lis[]用于保存第1~i元素元素中最长不下降序列的长度，则lis[i]=max(lis[j])+1,且num[i]>num[j],i>j。然后在lis[]中找到最大的一个值，时间复杂度是O(n^2)。

         代码实现：

int Longest\_Increasing(int num[],int n)

{

    int lis[n],i,j,max=0;

    for(i=0;i<n;i++)

       {

       lis[i]=1;

       for(j=0;j<i;j++)

       if (num[i]>num[j]&&lis[j]+1>lis[i])

           lis[i]=lis[j]+1;

       }

    for(i=0;i<n;i++)

       if(max<lis[i]) max=lis[i];

    return max;

}

二、复杂点的O(nlogn)算法

               概述：O(nlogn)的算法关键是它建立了一个数组b[]，b[i]表示长度为i的不下降序列中结尾元素的最小值，用K表示数组目前的长度，算法完成后K的值即为最长不下降子序列的长度。

               具体点来讲：

                设当前的以求出的长度为K，则判断a[i]和b[k]：

                1.如果a[i]>=b[k]，即a[i]大于长度为K的序列中的最后一个元素，这样就可以使序列的长度增加1，即K=K+1,然后现在的b[k]=a[i]；

                 2.如果a[i]<b[k]，那么就在b[1]...b[k]中找到最大的j,使得b[j]<a[i],然后因为b[j]<a[i]，所以a[i]大于长度为j的序列的最后一个元素，那么就可以更新长度为j+1的序列的最后一个元素，即b[j+1]=a[i]。

                 算法复杂度的分析：

               因为共有n个元素要进行计算；每次计算又要查找n次，所以复杂度是O(n^2)，但是，注意到b[]数组里的元素的单调递增的，所以我们可以用二分法，查找变成了logn次。这样算法的复杂度就变成了O(nlogn)。具体算法实现请看代码（7-13update：以前的blog用不了了，所以重新弄过了）。

               下面这段代码解决的是一道OI的题。

                      http://www.rqnoj.cn/Problem\_Show.asp?PID=167

            #include<iostream>

            using namespace std;

            long f[100001]={0},l=1,r,m,t=0,a;

            inline void BinarySearch(){

                     while(l<=r){

                           m=(l+r)>>1;

                           if(f[m]==a){l=m;return;}

                           else

                                  if(f[m]>a)l=m+1;

                                 else r=m-1;

                     }

             }

             main(){

                       long n;

                       cin>>n;

                       for(int i=1;i<=n;i++){

                            cin>>a;

                            if(a==0)continue;

                            l=1,r=t;

                            BinarySearch();

          if(l<=t)f[l]=a;

                            else t++,f[t]=a;

                       }

   cout<<t;

           }