L题题解报告

题意：求最长上升子序列（优化）

因为每次操作花费钱是固定的，所以我们要让操作最少

其实就是一个最长上升子序列，所有不在这个序列里的都是我们要操作的点。

求最长上升子序列，首先要把全部序列预处理一番，把问题转化成求最长不下降子序列。

具体就是，假如 **a[i]** 第i个数的大小，那么我们只需要

让 **a[i] = a[i] - i**;

思考一下为什么可以这样操作

假如 **a1 = 1, a2 = 2, a3 = 3……**

我们可以称这个序列为“条件上升子序列”（类比微积分绝对收敛、条件收敛）

然后把他们所有点都减去自己的下标

a1 = 0, a2 = 0, a3 = 0……

变成了“条件不下降子序列”

很奇妙，不过只可意会，难以言传。

现在问题转化成了求最长不下降子序列

朴素做法， **dp[i]** 表示以第i个数为终点的最长不下降子序列长度

转移方程

```
dp[i] = max (dp[i], dp[j]+1);
// 1 <= j <= i - 1 && arr[i] >= arr[j]
```

但是时间复杂度O(n^2)，不是很理想；

考虑到我们只需要知道最长不下降子序列的长度，

对最长子序列有哪些元素并不在意，

所以我们可以用另一个更快的求伪最长不下降子序列的方法（长度一样，但不是真正的最长不下降子序列）

其实就是结合了贪心的思想，为了让序列更长，

我们要尽量挑选比较小的数字；

具体来说，就是另开一个数组比如b[]

我们遍历到a[]的第i个数a[i]时，如果当前数比b[]的末尾大就直接插入到b[]的末尾，

否则就找到b[]中第一个大于a[i]的数然后用a[i]代替它。

注意，代替的时候我们并没有增加b[]的长度，

而是补充了b[]长度取得最大的“可能”（不知道怎么具体表示这个）

总之就是这个方法可以求最长不下降子序列的长度。

找“第一个大于a[i]的数”的时候可以用 **lower_bound()** 优化。

```
for(int i=1;i<=n;++i){
    scanf("%d",&arr[i]);
    arr[i]-=i;
    if(arr[i]<arr2[ans]){
        int tmp=lower_bound(arr2+1,arr2+ans+1,arr[i])-arr2;//找到符合要求的那个点
        arr2[tmp]=arr[i];//更新
        continue;
    }
    arr2[++ans]=arr[i];//比末尾元素还大，插入
}
```

> **时间复杂度：O（nlogn)**
>
> 从第一个点遍历到最后一个点，每遍历到一个点的时候利用lower_bound（）找点，单次logn