方法.

对给定的序列作一个排序（从小到大），要求在这个过程中删去重复元素. 例如 "1,3,2,5,2" $\xrightarrow{\text{排序}}$ "1,2,3,5"

然后找排序前后的两个序列的 LCS 即可

如果只要 $\Theta(n^2)$ 的话…

前面的排序使用简单排序就好.

排序复杂度为 $\Theta(n^2)$

```
b[0] = a[0], size_b = 1
for i=1 to n //n为a的长度
    for j= size_b-1 to 0
        t = a[i]
        if b[j]>t
            b[j+1]=b[j]
        else if b[j]==t
            break;
        else b[j+1]=t
            ++size_b;
            break;
```

相等的话 ← 就不写入 b 了.

现在得到了 a. b 两个数组，找 a, b 的 LCS.

方法同课上一致 (设 a 长为 n, b 长为 m)

	a	a[0]	a[1]	…	a[n-1]
b	0	0	0	…	0
b[0]	0				
⋮	⋮				
b[m-1]	0				

复杂度为 $\Theta(m \cdot n)$. $(m \le n)$

~~std::vector<std::vector<int>>~~

构建 L[m+1][n+1], 初始为零矩阵

```
for i=1 ; i≤n; ++i
    for j=1 ; j≤m; ++j
        if a[i-1]==b[j-1]
            L[i][j] = L[i-1][j-1] +1
        else
            L[i][j] = max{L[i-1][j],
                          L[i][j-1]}
```

L[i][j] 保存的是 a[0…i-1] 与 b[0…j-1] 的 LCS 长度

方法：

~~对路径遍历~~

得到 L 之后，找出 LCS.

一个最直接的办法是沿着构建L的路径回溯，假定用 lcs 来贮存结果

复杂度为 ~~$\Theta(m+n-1)$~~ $O(n)$.
(因为每一次 i 与 j 至少有一个减1，总减 m+n-1 次)

∴ 总复杂度 $\Theta(n^2)$.

```
index = max{L[m+1][n+1]}
i = n, j = m
while (i>0 and j>0)
    if a[i-1] == b[j-1]
        lcs[index-1] = a[i-1];
        --i;
        --j;
        --index
    else if L[j-1][i] > L[j][i-1]
            --j;
        else
            --i;
```

例子：

找 数列 序列 3,2,3,1,5 的一个严格递增子序列

① "排序" ⟶ 1,2,3,5

② 找 3,2,3,1,5 与 1,2,3,5 的 LCS

(1) L:

	a	3	2	3	1	5
b	0	0	0	0	0	0
1	0	0	0	0	1	1
2	0	0	1	1	1	1
3	0	1	1	2	2	2
5	0	1	1	2	2	3

(2) 找出 LCS (序列从0记起，即 a[0]=3)

∵ a[4] = b[3] = 5

a[3] = 1 ≠ b[2] = 3.

∵ ~~b[2]比b[1]~~ 2 > 1.

∴ 比较 a[2] 与 b[2]，相等，为3

a[1] = b[1]，相等，为2

a[0] ≠ b[0]，结束循环

故为 2, 3, 5.