第7章　排序

>>观看视频，回答问题

排序前（56, 34, 47, 23, 66, 18, 82, 47）

若排序后得到结果

（18, 23, 34, 47, 47, 56, 66, 82）

则称该排序方法是不是稳定的?

一、简单排序思想

1、冒泡排序 >>观看视频

9 8 5 4 2 0 如何排序?

8 5 4 2 0 9
5 4 2 0 8 9
4 2 0 5 8 9
2 0 4 5 8 9
0 2 4 5 8 9

【练习】使用冒泡排序法实现

9 1 5 8 3 7 4 6 2

的排序，要求写出每一趟的结果

2、插入排序 >>观看视频

算法思想：教材P268

9 8 5 4 2 0 如何排序?

8 9 5 4 2 0
5 8 9 4 2 0
4 5 8 9 2 0
2 4 5 8 9 0
2 4 5 8 9 0

【练习】使用插入排序法实现

9 1 5 8 3 7 4 6 2

的排序，要求写出每一趟的结果

3、选择排序

9 8 5 4 2 0 如何排序?

0 8 5 4 2 9
0 2 5 4 8 9
0 2 4 5 8 9
0 2 4 5 8 9
0 2 4 5 8 9

【练习】使用选择排序法实现

9 1 5 8 3 7 4 6 2

的排序，要求写出每一趟的结果

【课堂作业】 P288 7.5

二、简单排序算法实现

```
void X_Sort ( ElementType A[], int N )
```

void x_Sort(SqList *L)

为了与之前学的内容更好地衔接上，
我们这里讨论的排序数据都存储在顺序表中。

```
#define MAXSIZE  20
typedef struct
{
        int r[MAXSIZE+1];
        int length;
}SqList;
```

1、冒泡排序

```
#define MAXSIZE  20
typedef struct
{
      int r[MAXSIZE+1];
      int length;
}SqList;
```

```
void BubbleSort(SqList *L)
{
    int i,p,t=0;   //p是最后一个数在的位置
    for(p=L->length;p>=1;p--)
    for(i=1;i<p;i++)
        if (L->r [i] >L->r [i+1] )
        {
         t=L->r[i];L->r[i]=L->r[i+1];
         L->r [i+1] =t;
        }
}
```

这个算法效率是否是最高的？它存在什么问题？

若需要排序的数字是：９１２３４

程序的执行过程是怎样的？

```
//改进的冒泡排序
void BubbleSort2(SqList *L)
{
    int i,p,t=0;
    int flag;
    for(p=L->length;p>=1;p--)
    {
      flag=0;
      for(i=1;i<p;i++)
        if (L->r [i] >L->r [i+1] )
        {
         t=L->r[i];L->r[i]=L->r[i+1];
         L->r [i+1] =t;flag=1;
        }
     if(flag==0)  break;
    }
}
```

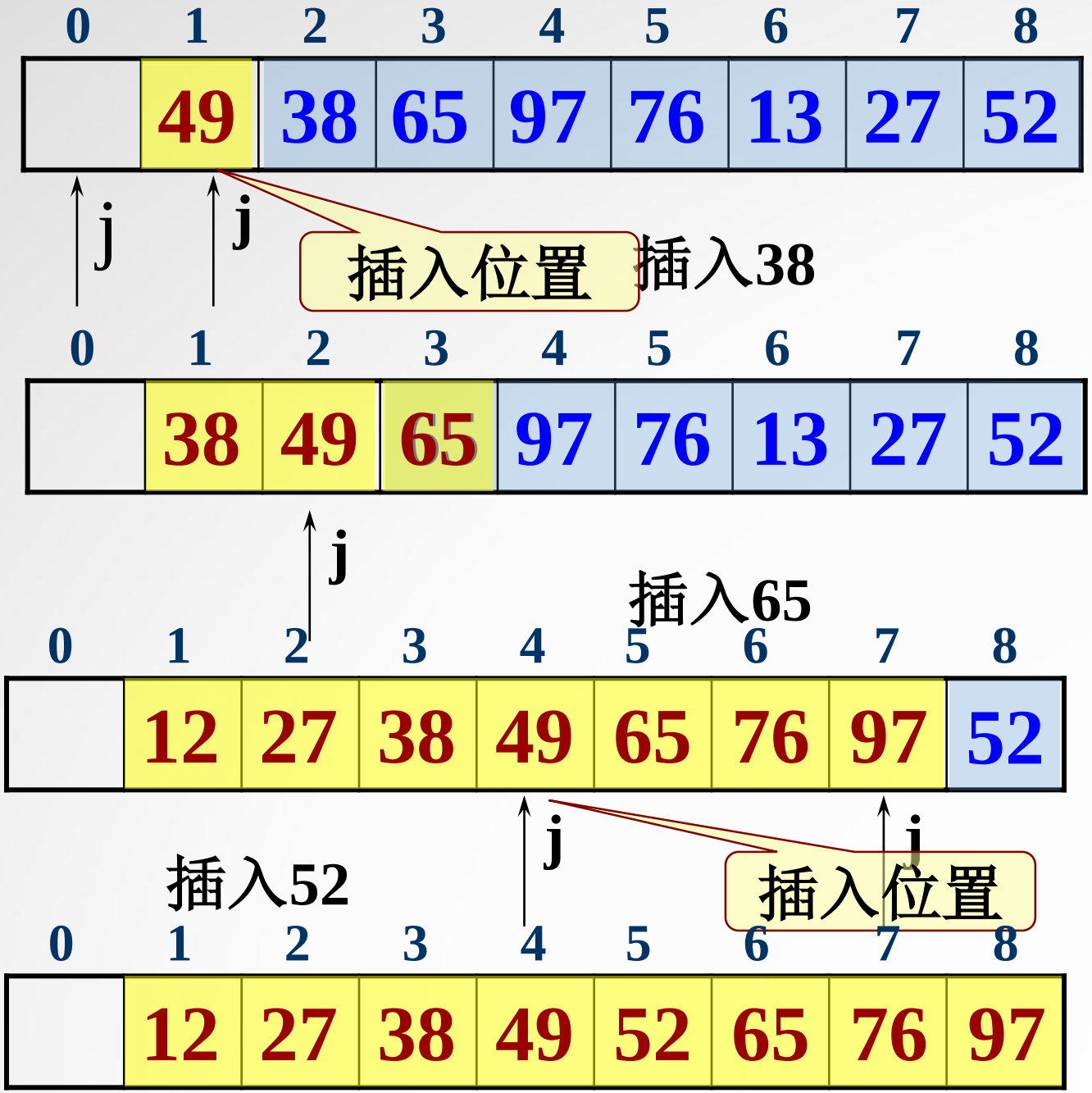
- 算法复杂度分析（教材P272）

  最好的情况，顺序有序O(n)
  最坏的情况，逆序有序$O(n^2)$

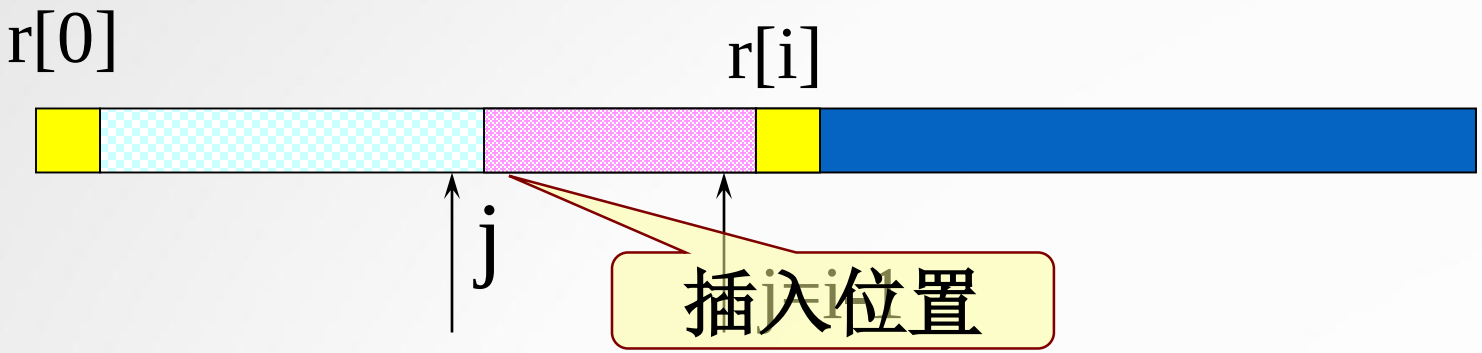
- 算法稳定性分析

  稳定

2、插入排序


0 1 2 3 4 5 6 7 8
49 38 65 97 76 13 27 52
j
j
插入位置
插入38
0 1 2 3 4 5 6 7 8
38 49 65 97 76 13 27 52
j
插入65
0 1 2 3 4 5 6 7 8
12 27 38 49 65 76 97 52
j
j
插入52
插入位置
0 1 2 3 4 5 6 7 8
12 27 38 49 52 65 76 97

```
for ( i=2;  i<=n; i++ )
if (r[i]<r[i-1])
{ 利用顺序查找实现在 R[1..i-1]中查找R[i]的插入位置;
  插入R[i] ;}
```




```
    r[0] = r[i];                    // 设置“哨兵”
for (j=i-1; r[0]<r[j]; j--) // 从后往前找，顺带“腾位置”
        r[j+1] = r[j];
        r[j+1] = r[0]
```

```
void InsertSort (SqList *L)
{
    int i,j;
    for(i=2;i<=L->length;i++ )
   {
     if(L->r[i]<L->r[i-1])
     {
        L->r[0]=L->r[i];
        for(j=i-1;L->r[j]>L->r[0];j--)
              L->r[j+1]=L->r[j];
        L->r[j+1]=L->r[0];
      }
    }
}
```

➢算法复杂度分析（教材P269）

最好的情况，顺序有序O(n)
最坏的情况，逆序有序$O(n^2)$

➢算法稳定性分析

稳定

3、选择排序

```
void SelectSort (SqList *L)
{
    int i,j,min,t=0;
    for(i=1;i<L->length;i++ )
   {
     min=i;
     for(j=i+1;j<=L->length;j++ )
         if(L->r[min]>L->r[j]) min=j;
     if(i!=min)
      {t=L->r[i];L->r[i]=L->r[min];L->r[min]=t;}
     }
}
```

➢算法复杂度分析（教材P265）

$O(n^2)$，无所谓最小还是最坏

➢算法稳定性分析

如，待排序列为 **5** 8 5 2 9

不稳定