



哈尔滨工业大学
HARBIN INSTITUTE OF TECHNOLOGY

规格严格 功夫到家



第7章 数组

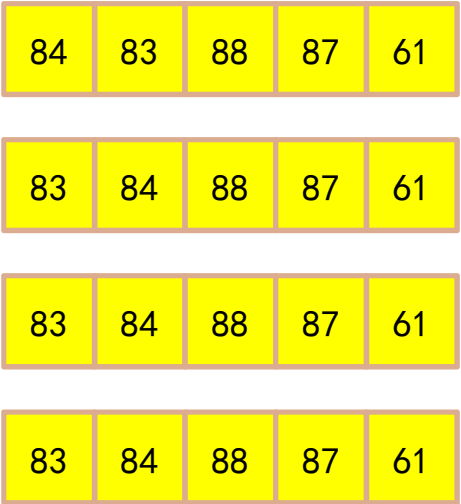
——排序算法的函数实现



哈尔滨工业大学
叶麟

交换法排序

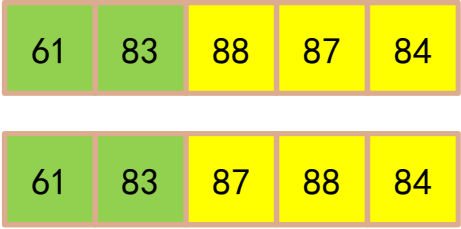
第1遍



第2遍



第3遍



第4遍



第一个数分别与后面所有的数进行比较，若后面的数较小，则交换后面这个数和第一个数的位置

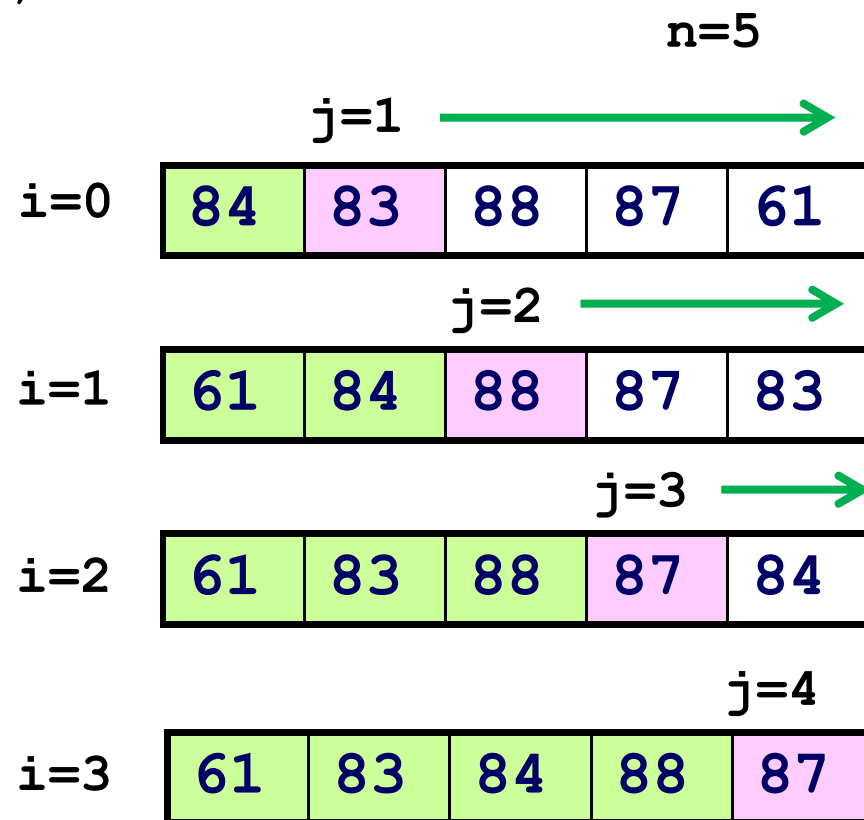
第二个数分别与后面所有的数进行比较，若后面的数较小，则交换后面这个数和第二个数的位置

.....

交换法排序

用**交换法**对成绩数组升序排序

```
int score[5] = {84, 83, 88, 87, 61};  
for (i=0; i<n-1; i++)  
{  
    for (j=i+1; j<n; j++)  
    {  
        if (score[j] < score[i])  
            "交换score[j]和score[i]"  
    }  
}
```



用交换法对成绩数组升序排序

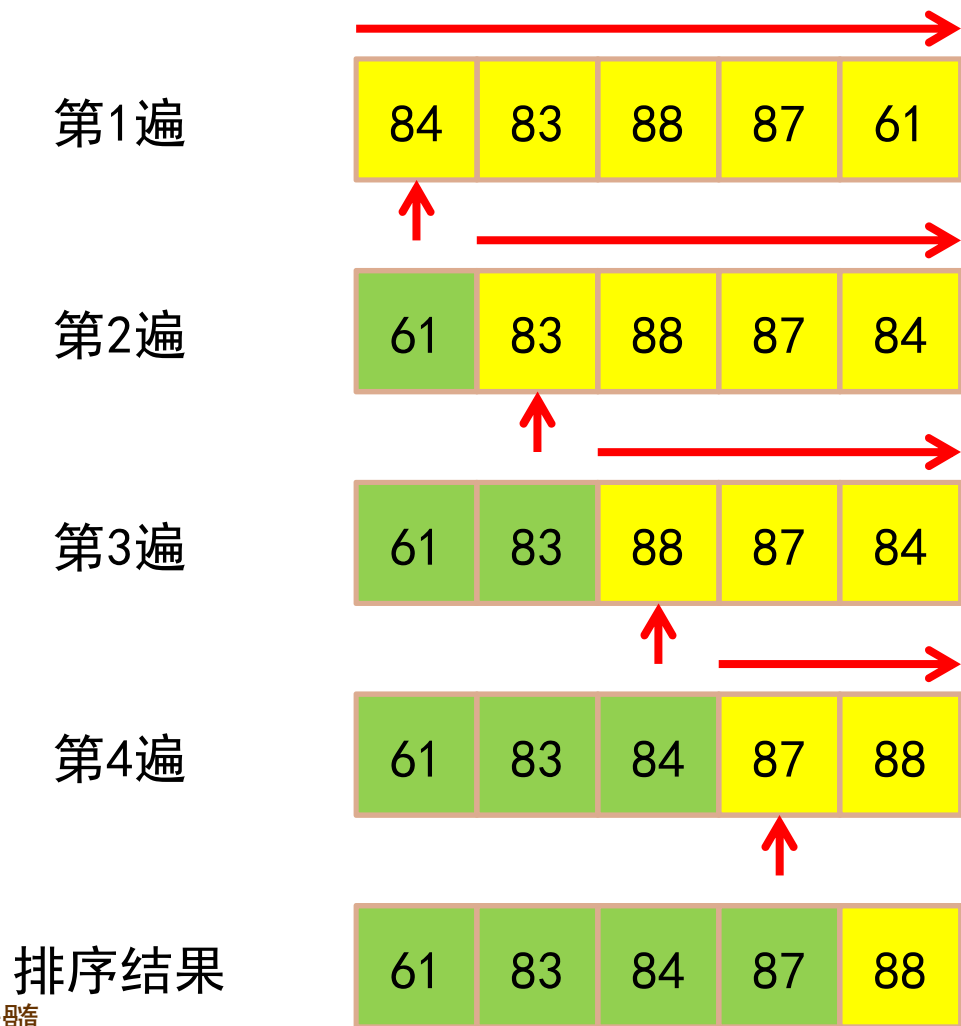
```
void ChangeSort(int score[], int n) /*交换法排序*/
{
    int i, j, temp;
    for (i=0; i<n-1; i++)
    {
        for (j=i+1; j<n; j++)
        {
            if (score[j] < score[i]) /*从低到高*/
            {
                temp = score[j];
                score[j] = score[i];
                score[i] = temp;
            }
        }
    }
}
```

成绩升序排序

有改进的方法吗？



选择法排序



在每一遍比较中，在剩余的待比较的数中选择一个最小的数与这个剩余序列的第1个数交换位置

选择法排序

```
for (i=0; i<n-1; i++)  
{
```

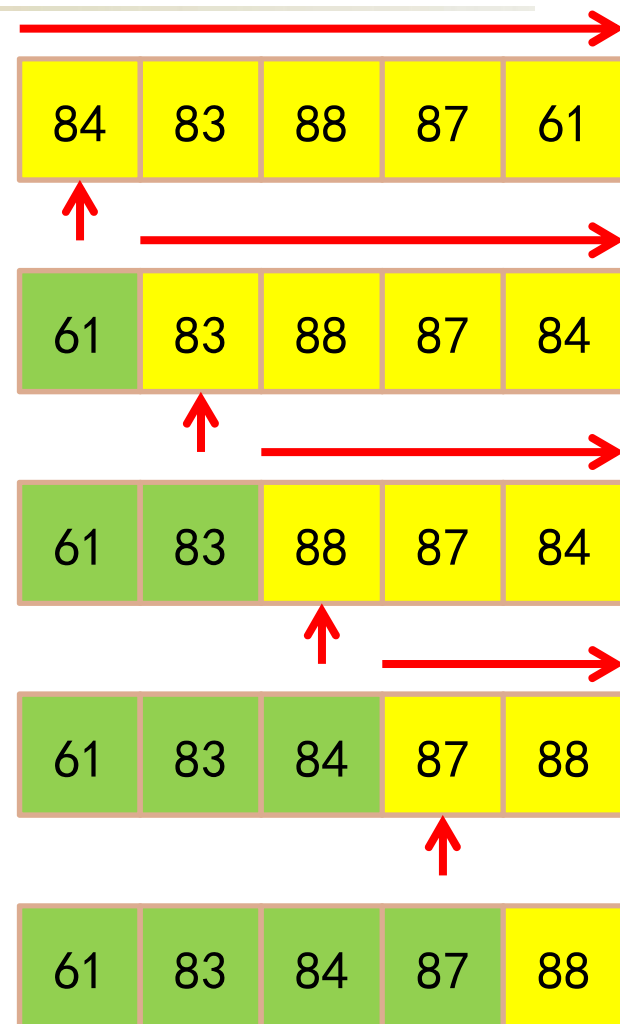
寻找最低分所在下标k的过程

```
    k = i;  
    for (j=i+1; j<n; j++)  
    {  
        if (score[j] < score[k])  
            记录此轮比较中最低分  
            所在元素的下标 k = j;  
    }
```

若k中记录的最低分位置不在下标i处，则

"交换成绩score[k]和score[i]"

```
}
```



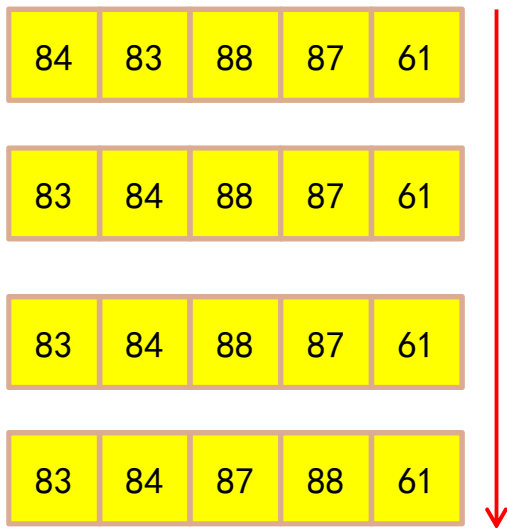
选择法排序

```
void SelectionSort(int score[], int n) /*选择法*/  
{  
    int i, j, k, temp;  
    for (i=0; i<n-1; i++)  
    {  
        k = i;  
        for (j=i+1; j<n; j++)  
        {  
            if (score[j] < score[k])  
            {  
                k = j; /*记录最小数下标位置*/  
            }  
        }  
        if (k != i) /*若最小数不在下标位置i*/  
        {  
            temp = score[k];  
            score[k] = score[i];  
            score[i] = temp;  
        }  
    }  
}
```

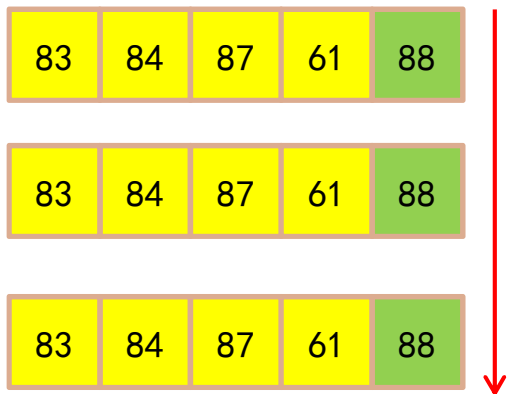
成绩升序排序

冒泡法排序

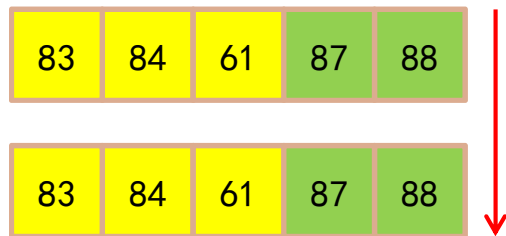
第1遍



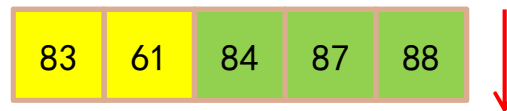
第2遍



第3遍



第4遍



比较相邻的两个数据
若顺序不对，则将其位置交换

冒泡法排序

```
void BubbleSort(int score[], int n)
{
    int i, j, temp;
    for (i=0; i<n-1; i++)
    {
        for (j=1; j<n-i; j++)
        {
            if (score[j] < score[j-1])
            {
                temp = score[j];
                score[j] = score[j-1];
                score[j-1] = temp;
            }
        }
    }
}
```

交换相邻元素

有改进的方法吗？



归并法排序

5	24	35	74	222
---	----	----	----	-----

19	23	30
----	----	----

归并法排序

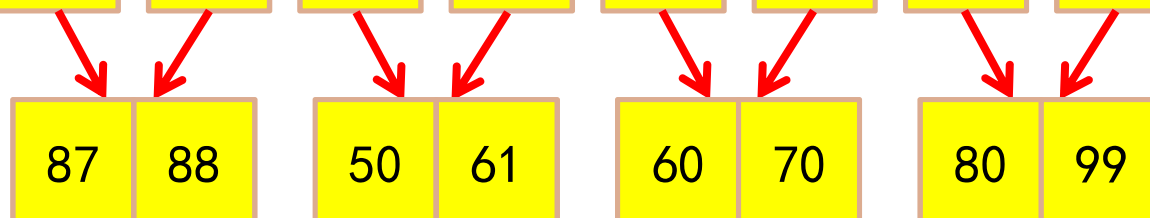
83	84	87	88	61	50	70	60	80	99
----	----	----	----	----	----	----	----	----	----

归并法排序

原数据



第1次合并后

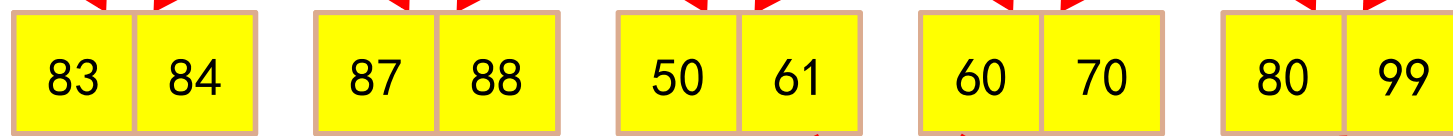


归并法排序

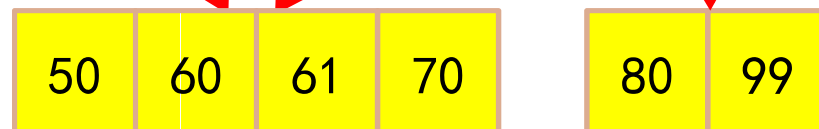
原数据



第1次合并后



第2次合并后

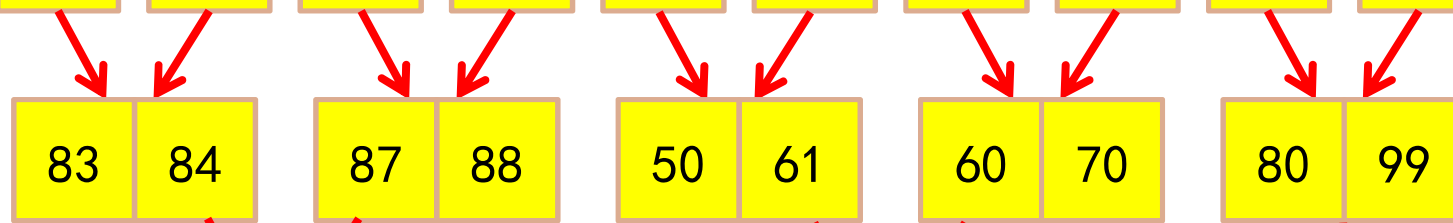


归并法排序

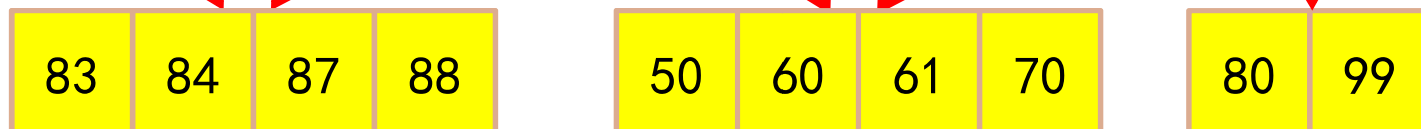
原数据



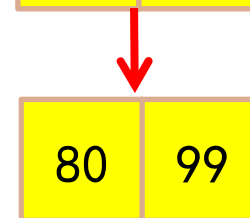
第1次合并后



第2次合并后

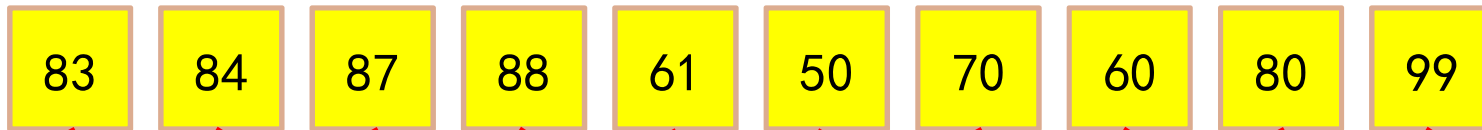


第3次合并后

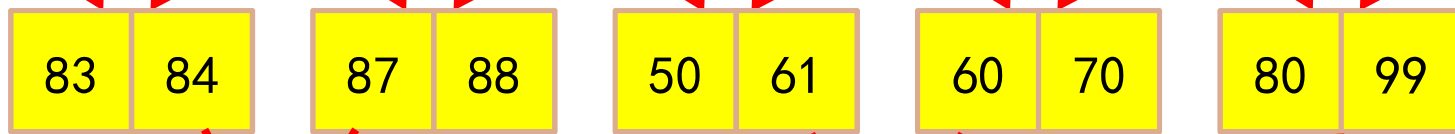


归并法排序

原数据



第1次合并后



第2次合并后



第3次合并后



第4遍合并后

归并法排序

```
void MergeSort(int score[], int start, int end)
{
    if (start < end)
    {
        int i;
        i = (end + start) / 2;
        MergeSort(score, start, i);
        MergeSort(score, i + 1, end);
        Merge(score, start, i, end);
    }
}
```

对前半部分进行排序

对后半部分进行排序

合并前后两部分

归并法排序

```
void Merge(int score[], int left, int m, int right)
```

```
{
```

```
    int aux[100] = {0};
```

```
    int i, j, k;
```

```
    // 分别将 i, j, k 指向各自的首部
```

```
    for (i = left, j = m+1, k = 0; k <= right-left; k++)
```

```
    {
```

```
        //若 i 到达第一个数组的尾部, 将第二个数组余下元素复制到临时数组中
```

```
        if (i == m+1)
```

```
        {    aux[k] = score[j++];    continue;    }
```

```
        //若 j 到达第二个数组的尾部, 将第一个数组余下元素复制到临时数组中
```

```
        if (j == right+1)
```

```
        {    aux[k] = score[i++];    continue;    }
```

```
        //如果第一个数组的当前元素比第二个数组的当前元素小, 将第一个数组的当前元素复制到临时数组中
```

```
        if (score[i] < score[j])
```

```
        {    aux[k] = score[i++]; }
```

```
        //如果第二个数组的当前元素比第一个数组的当前元素小, 将第二个数组的当前元素复制到临时数组中
```

```
        else
```

```
        {    aux[k] = score[j++]; }
```

```
    }
```

```
    //将有序的临时数组存入score中
```

```
    for (i = left, j = 0; i <= right; i++, j++)
```

```
    {    score[i] = aux[j];    }
```

```
}
```

