广州大学

GU CN Computer Networks Institute

数据结构

Data Structure

主讲人：王国军

计算机科学与网络工程学院

csgjwang@gzhu.edu.cn
http://trust.gzhu.edu.cn/

办公室：行政西楼前座532室

根据严蔚敏老师《数据结构》（C语言版）（第2版）制作，仅供广州大学计算机、软件工程、网络工程及相关专业2024级本科生和任课老师使用。

第10章 排序

GU CN
Computer Networks
Institute

第十章 排序

GU CN
Computer Networks
Institute

1. 概述

排序：将一组数据元素排列成一个按关键字有序的序列（升序或降序）。

GU CN
Computer Networks
Institute

1. 概述

排序：将一组数据元素排列成一个按关键字有序的序列（升序或降序）。

排序是计算机程序设计中的一种重要运算，是计算机处理数据时一种频繁要做的工作。

GU CN
Computer Networks
Institute

概述：排序分类

按待排序记录所在位置

- 内部排序：在内存中进行的排序。
- 外部排序：待排序元素的数量很大，以至于内存中一次不能容纳全部元素，在排序过程中需对外存进行访问的排序。

GU CN
Computer Networks
Institute

概述：排序分类

按待排序记录所在位置

- 内部排序：在内存中进行的排序。
- 外部排序：待排序元素的数量很大，以至于内存中一次不能容纳全部元素，在排序过程中需对外存进行访问的排序。

按排序依据原则

- 插入排序：直接插入排序、折半插入排序、希尔排序
- 交换排序：冒泡排序、快速排序
- 选择排序：简单选择排序、树形选择排序、堆排序
- 归并排序：2-路归并排序
- 分配排序：基数排序（借助“多关键字排序思想”）

GU CN
Computer Networks
Institute

概述：排序分类

按排序所需工作量

- 简单的排序方法：$T(n)=O(n^2)$
- 先进的排序方法：$T(n)=O(n\cdot\log_2 n)$
- 基数排序：$T(n)=O(d\cdot n)$　(d为关键字的位数)

GU CN
Computer Networks
Institute

概述：排序分类

按排序所需工作量

- 简单的排序方法：$T(n)=O(n^2)$
- 先进的排序方法：$T(n)=O(n\cdot\log_2 n)$
- 基数排序：$T(n)=O(d\cdot n)$　(d为关键字的位数)

排序的两个基本操作

- 比较两个关键字的大小
- 将记录从一个位置移动到另一个位置

GU CN
Computer Networks
Institute

概述：排序分类

按排序所需工作量

- 简单的排序方法：$T(n)=O(n^2)$
- 先进的排序方法：$T(n)=O(n\cdot\log_2 n)$
- 基数排序：$T(n)=O(d\cdot n)$　(d为关键字的位数)

排序的两个基本操作

- 比较两个关键字的大小
- 将记录从一个位置移动到另一个位置

评价一个排序方法好坏的标准

- 排序所需比较关键字的次数
- 排序所需移动记录的次数
- 排序过程中所需的辅助空间的大小

GU CN Computer Networks Institute

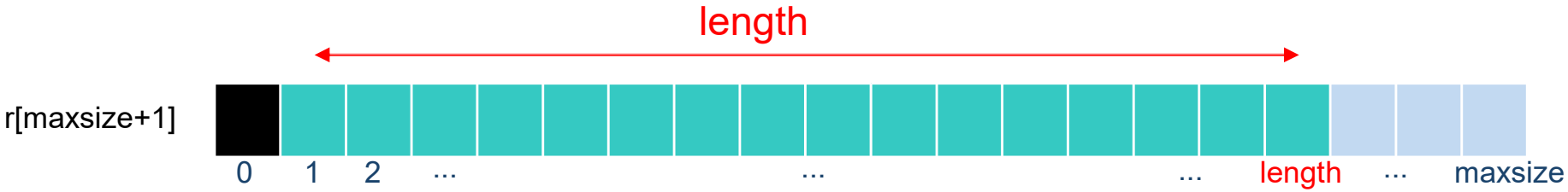
概述：排序的数据结构

```
typedef struct {
    KeyType  key;
    InfoType  otherinfo;
} ElemType;

typedef struct {
   ElemType r[maxsize+1]  // r[0]不放元素
   int length;
} SqList;
```

GU CN
Computer Networks
Institute

概述：排序的数据结构

```
typedef struct {
    KeyType  key;
    InfoType  otherinfo;
} ElemType;

typedef struct {
    ElemType r[maxsize+1]  // r[0]不放元素， 设置监视哨
    int length;
} SqList
```



GU CN
Computer Networks
Institute

2 插入排序

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

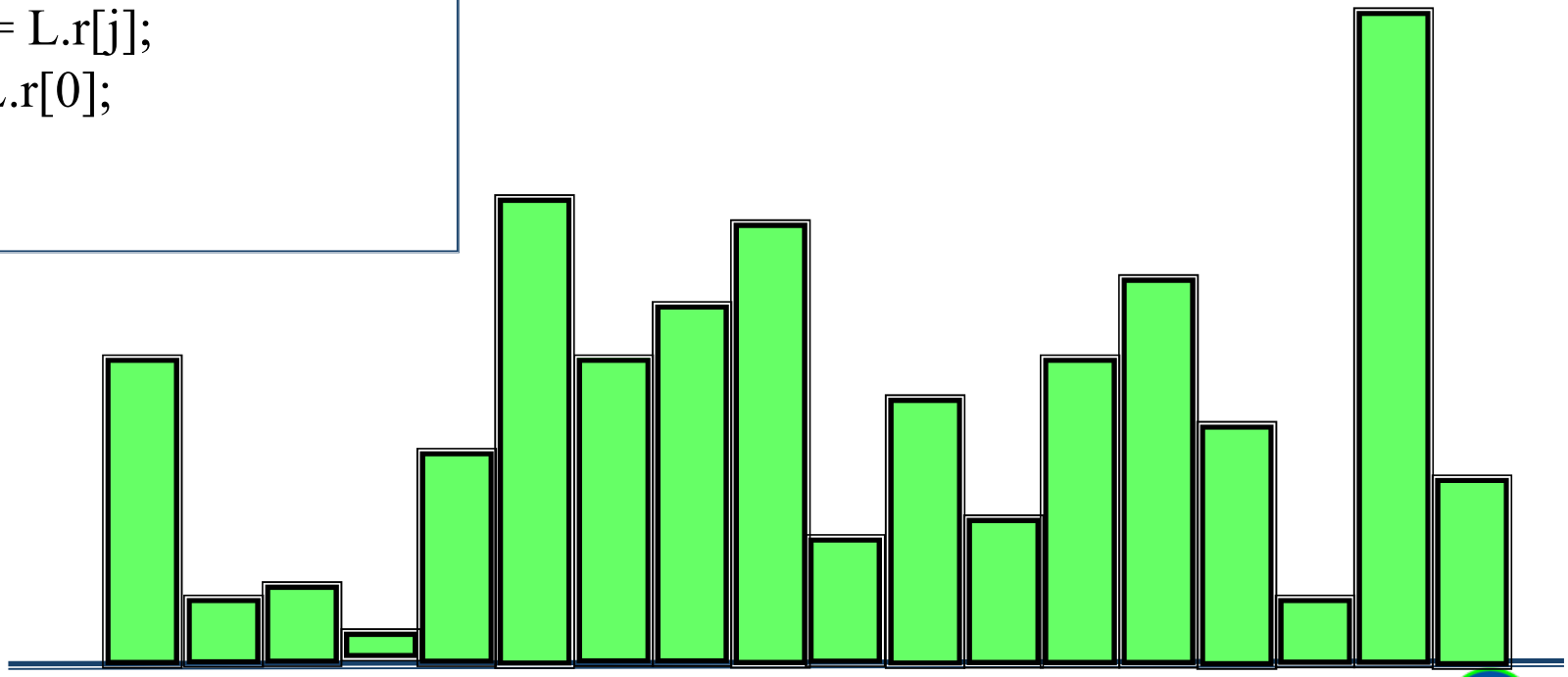
直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

GU CN Computer Networks Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

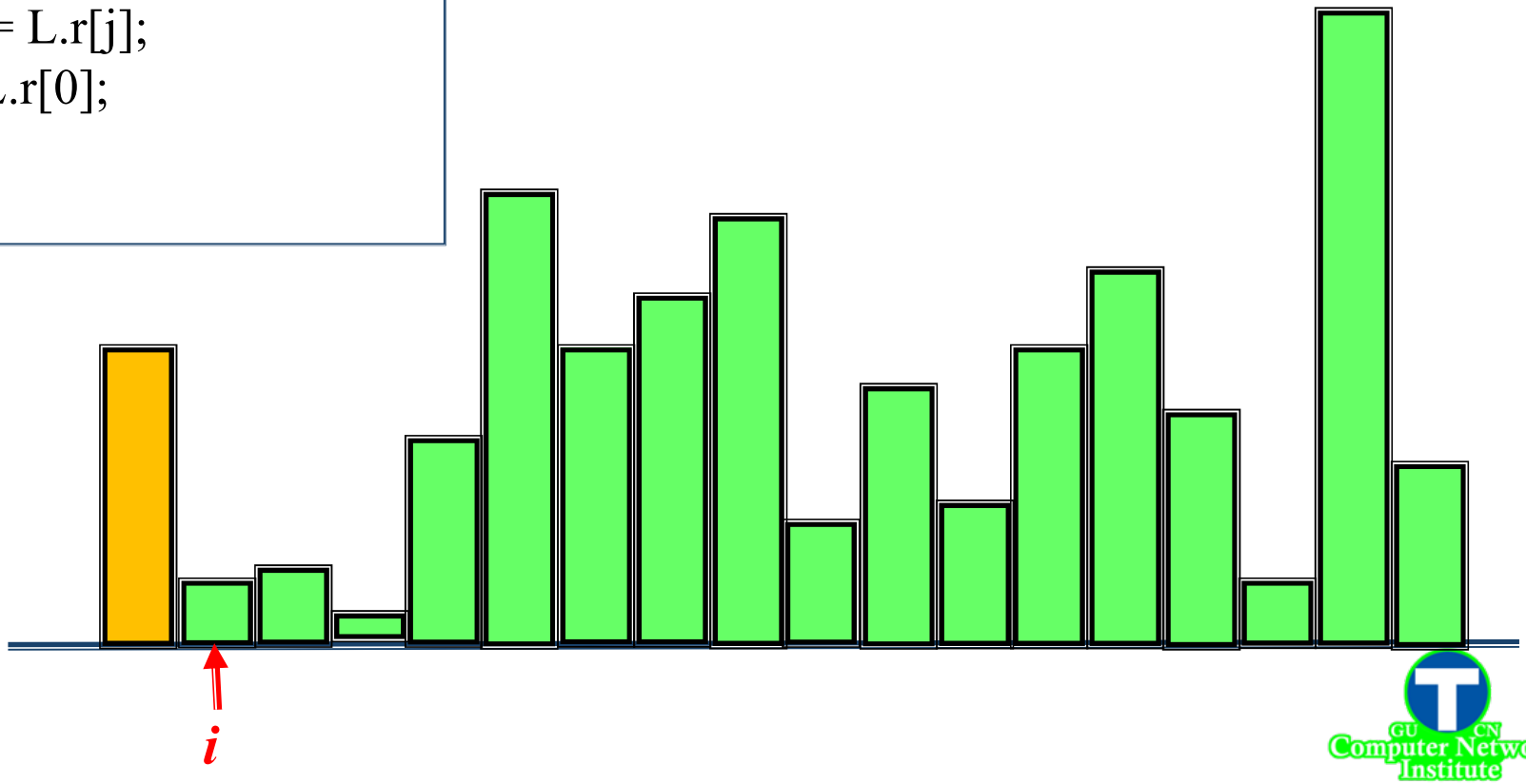


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

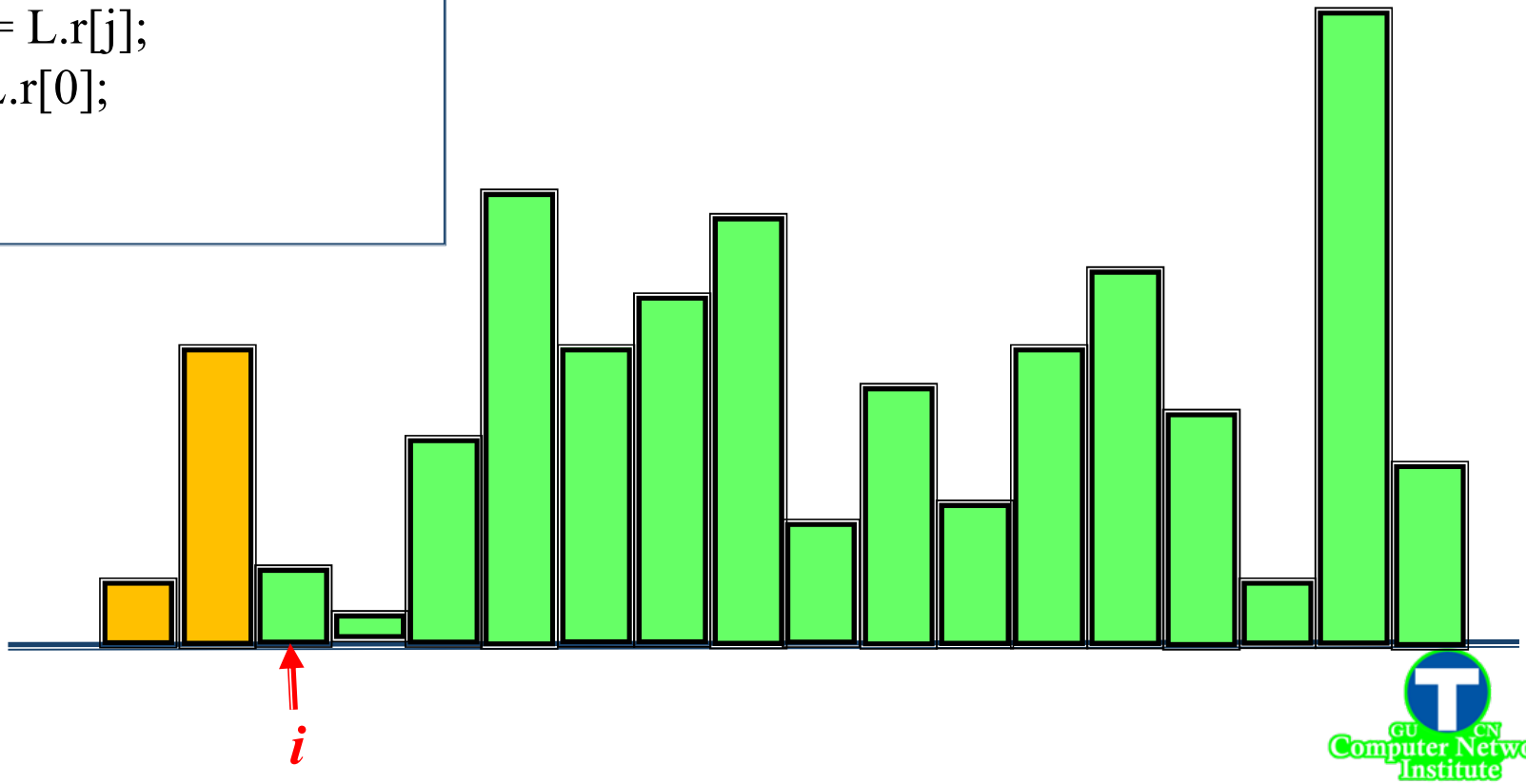


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

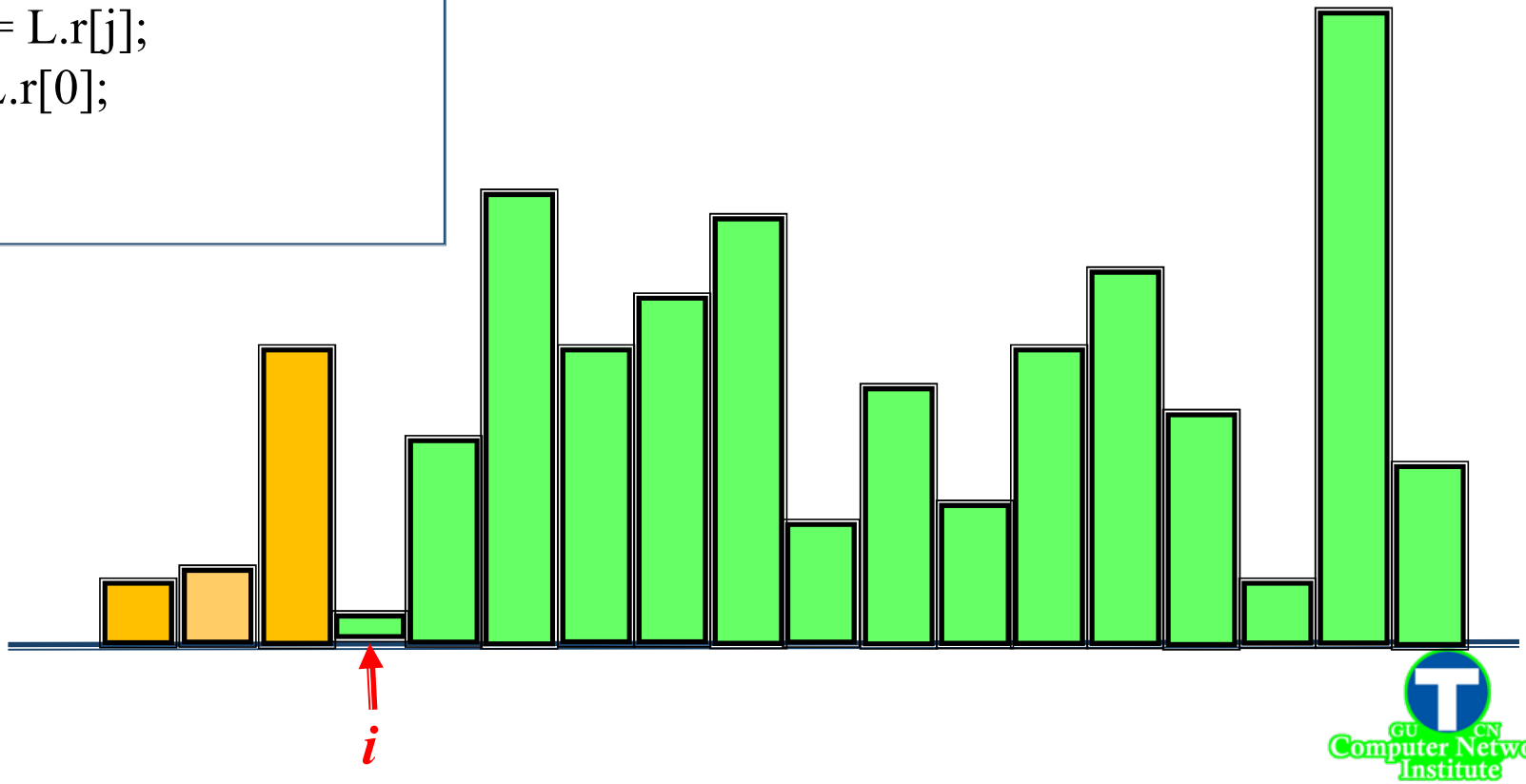


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

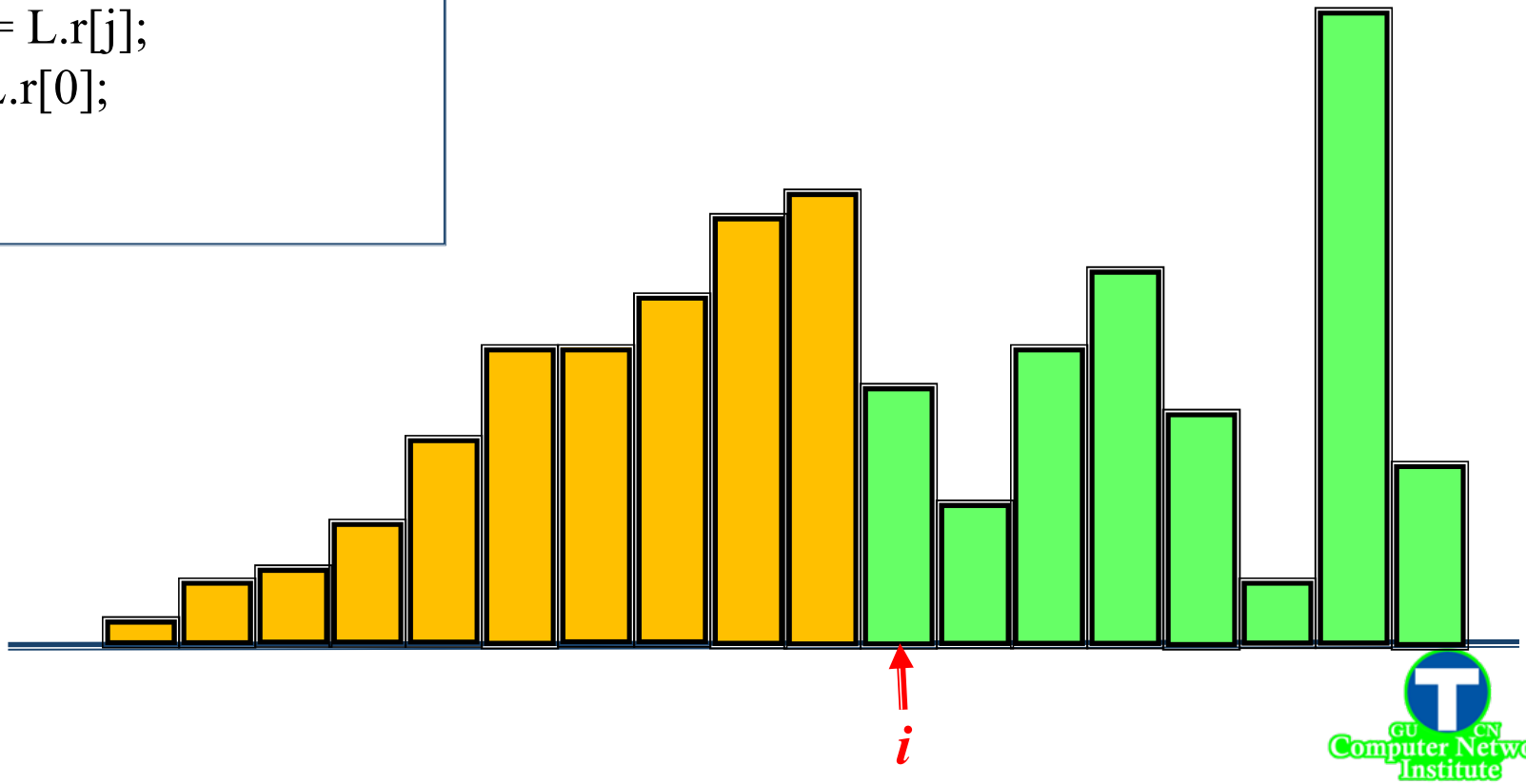


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

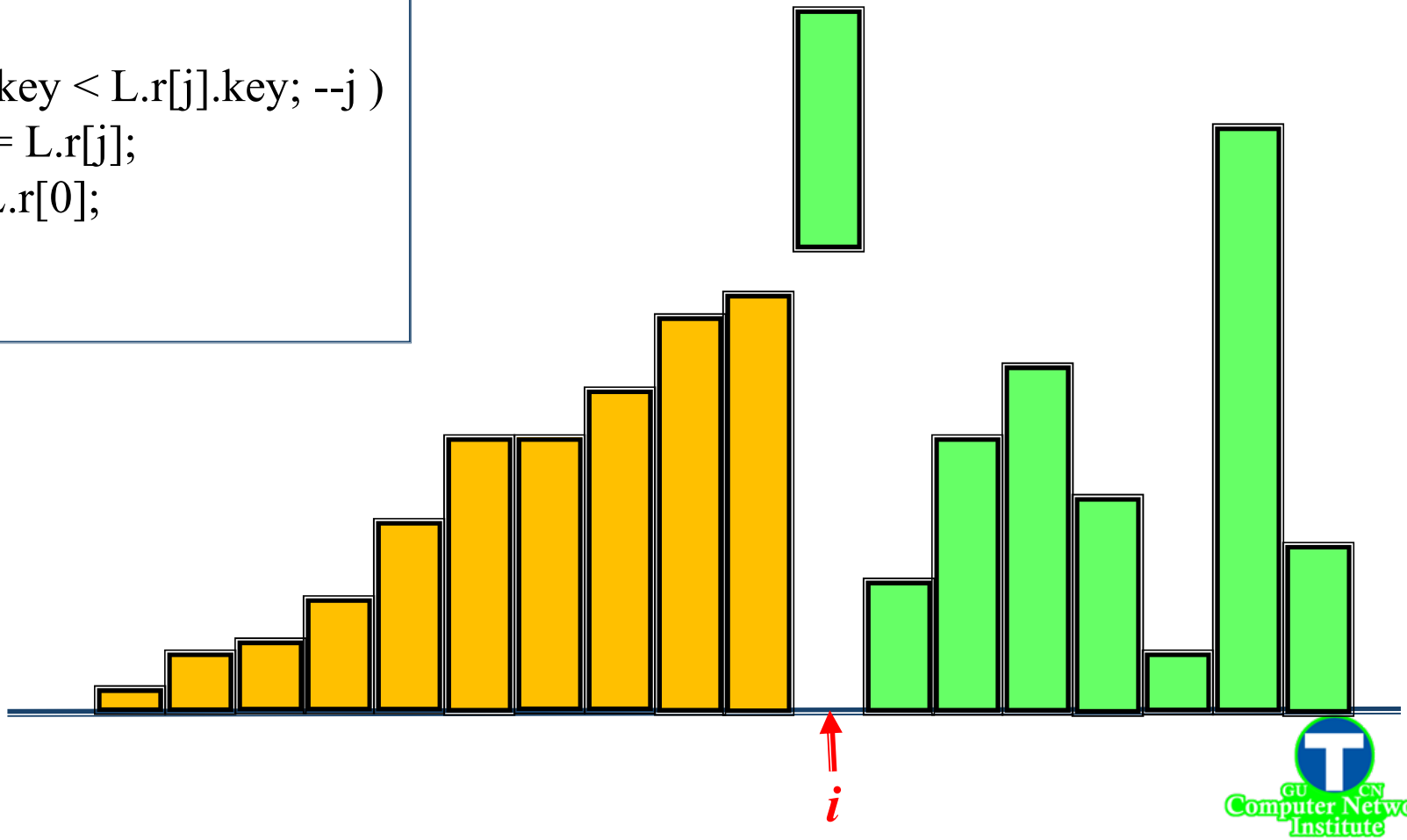


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

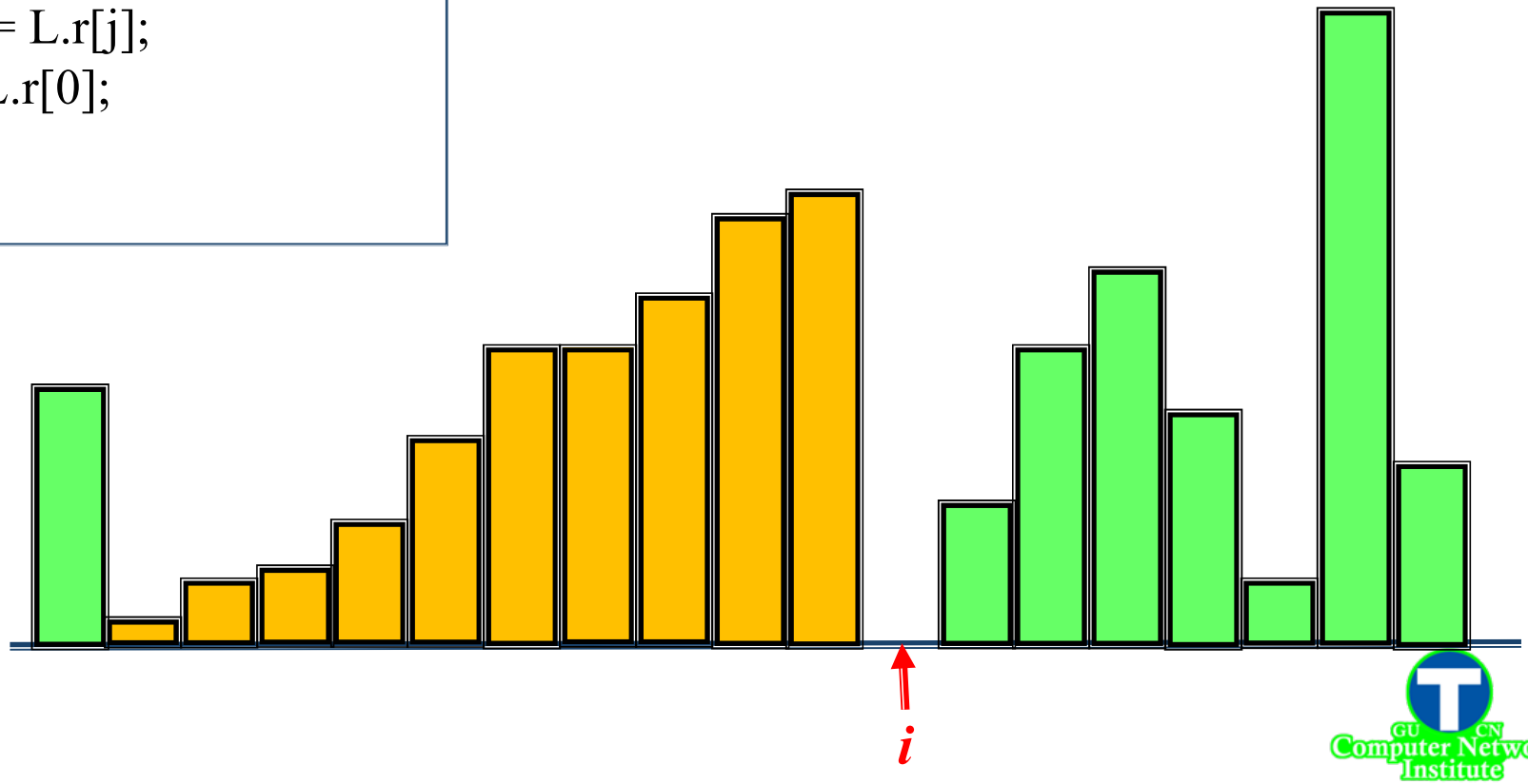
直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。



2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

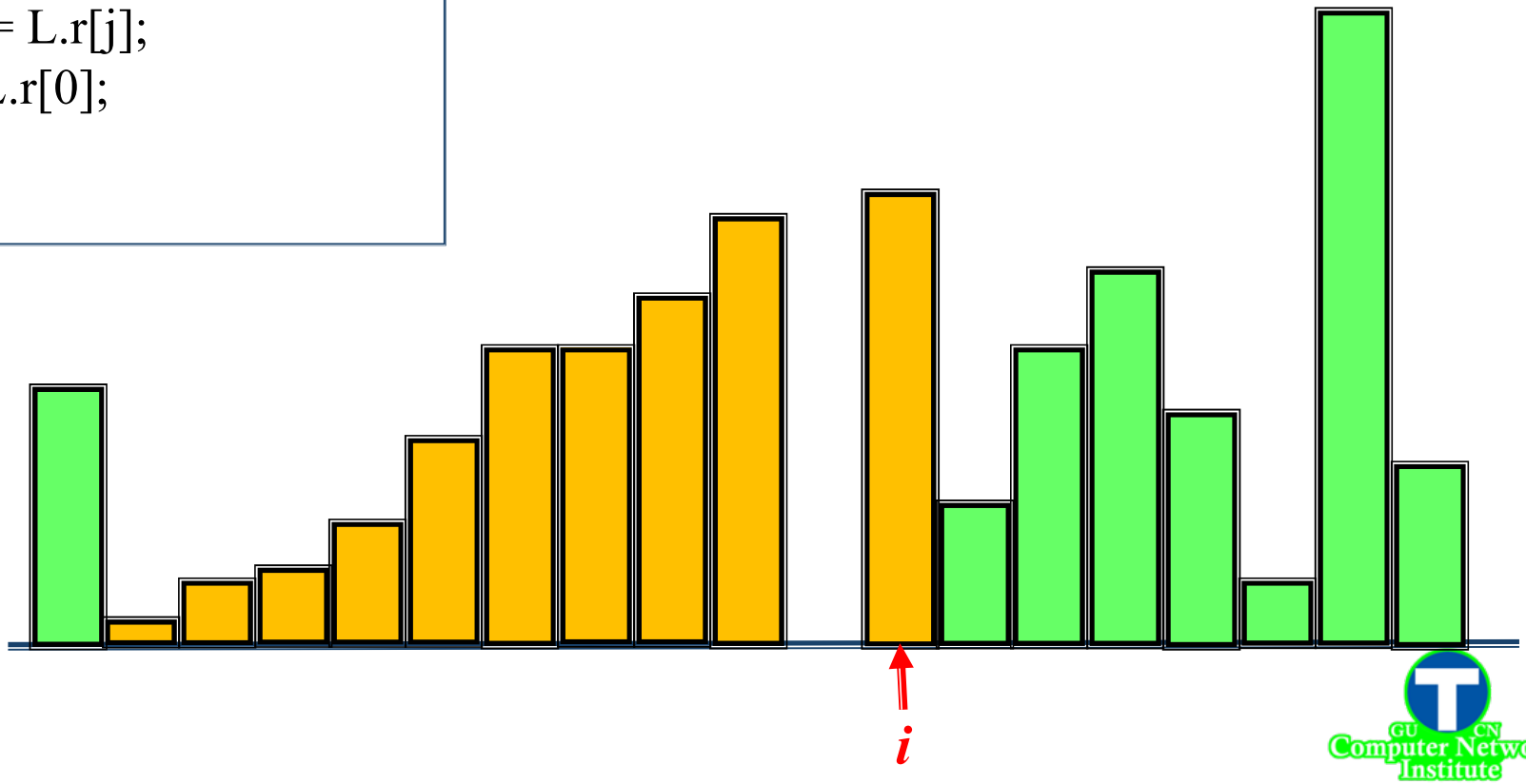


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

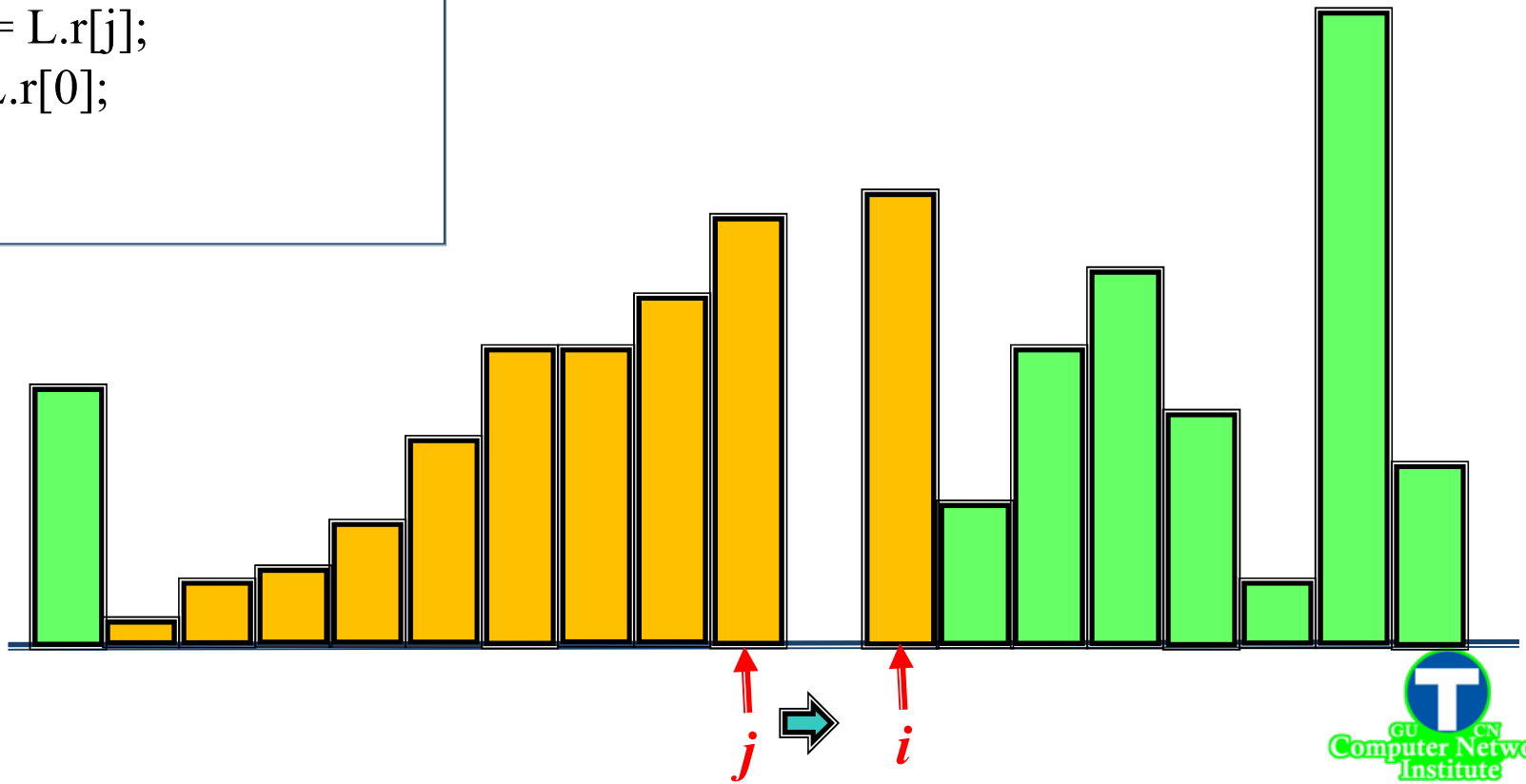
直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。



2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

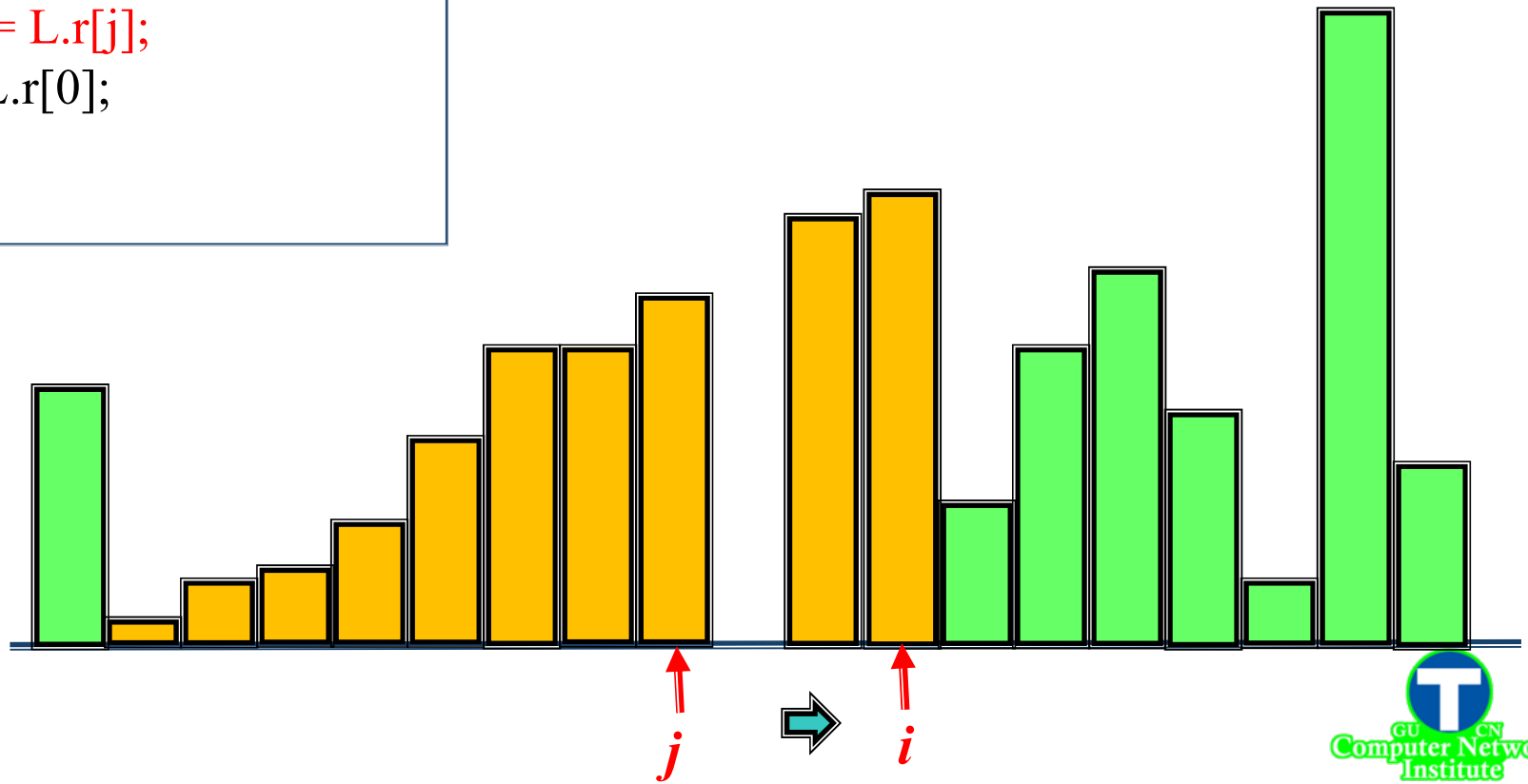


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

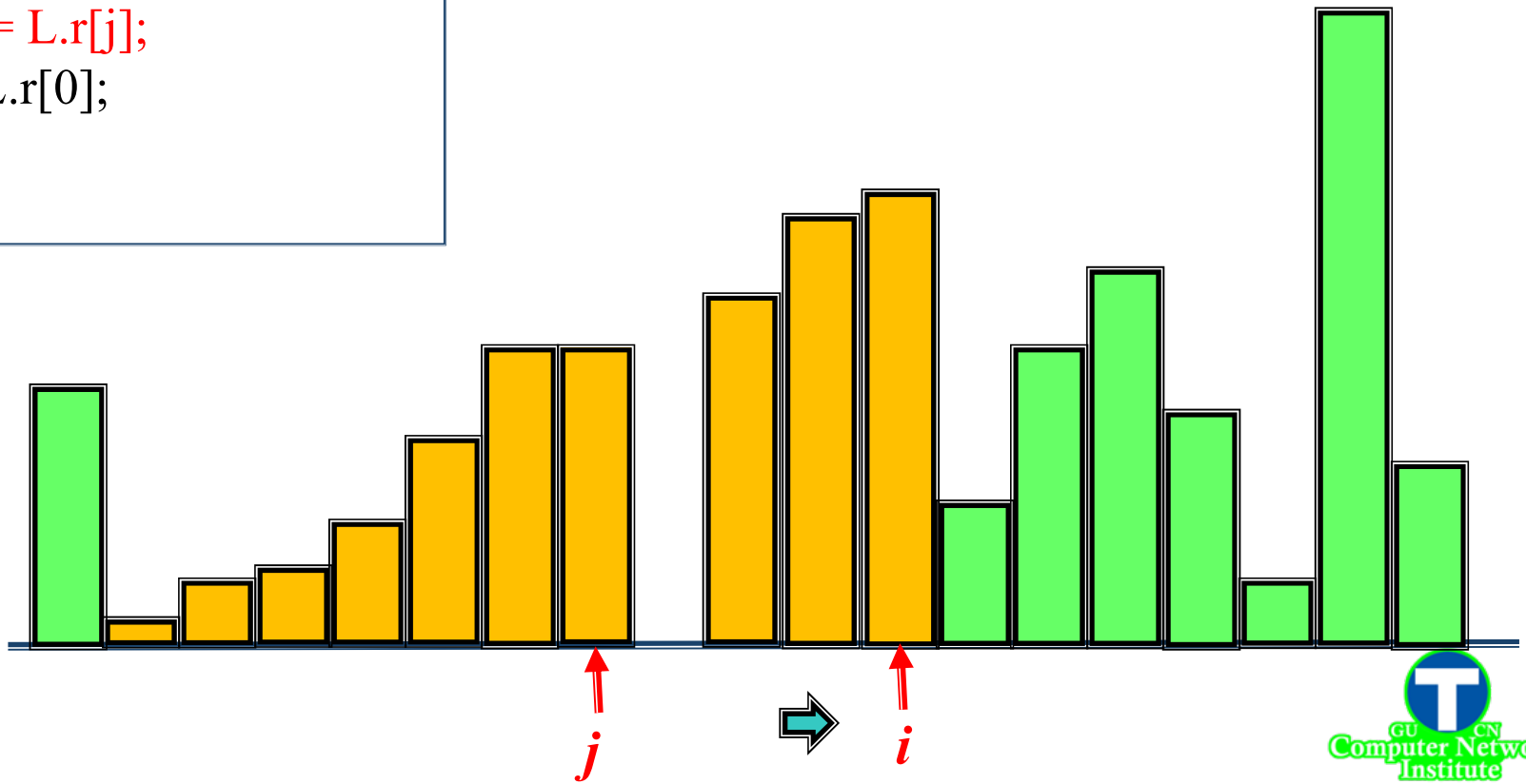


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

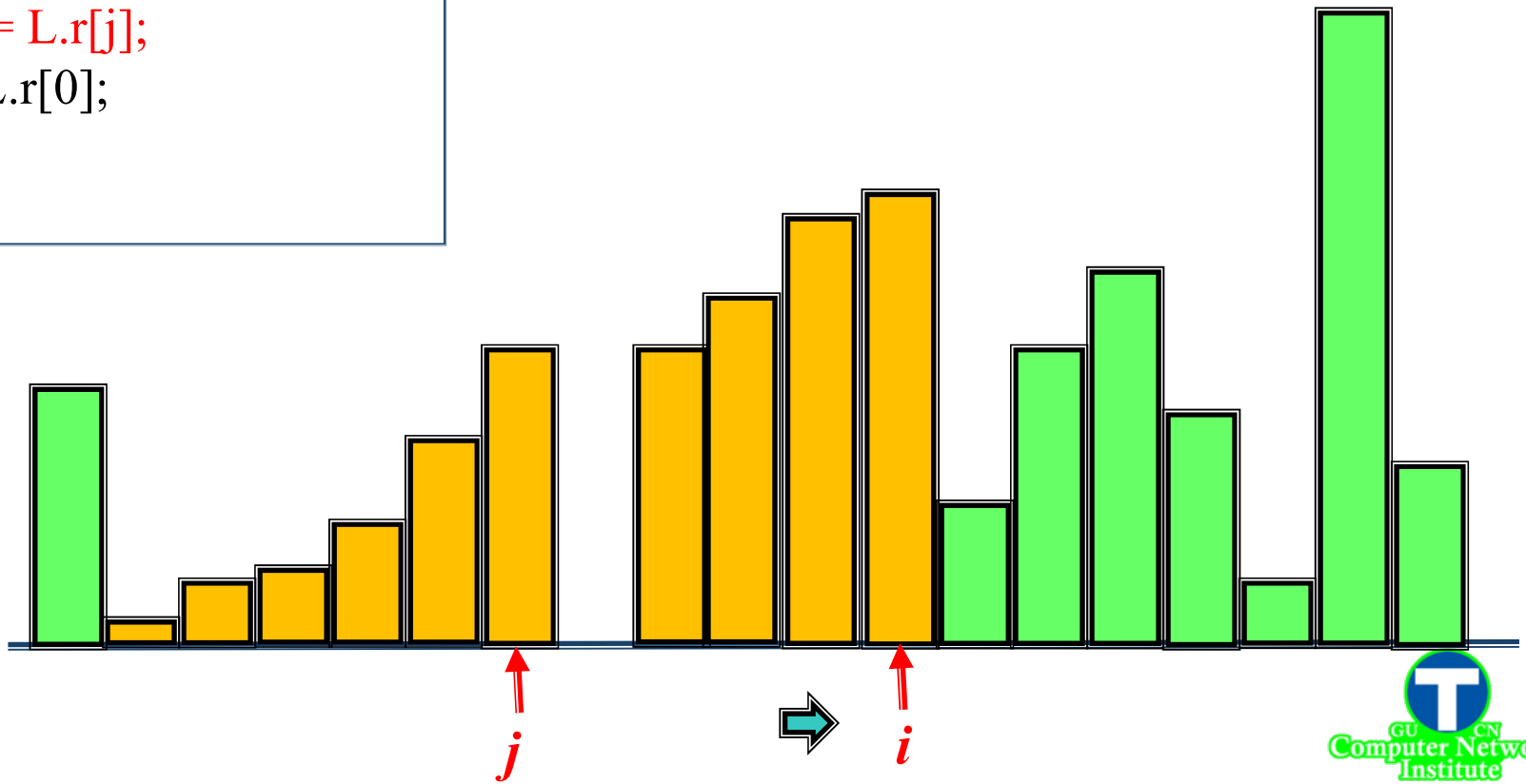


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

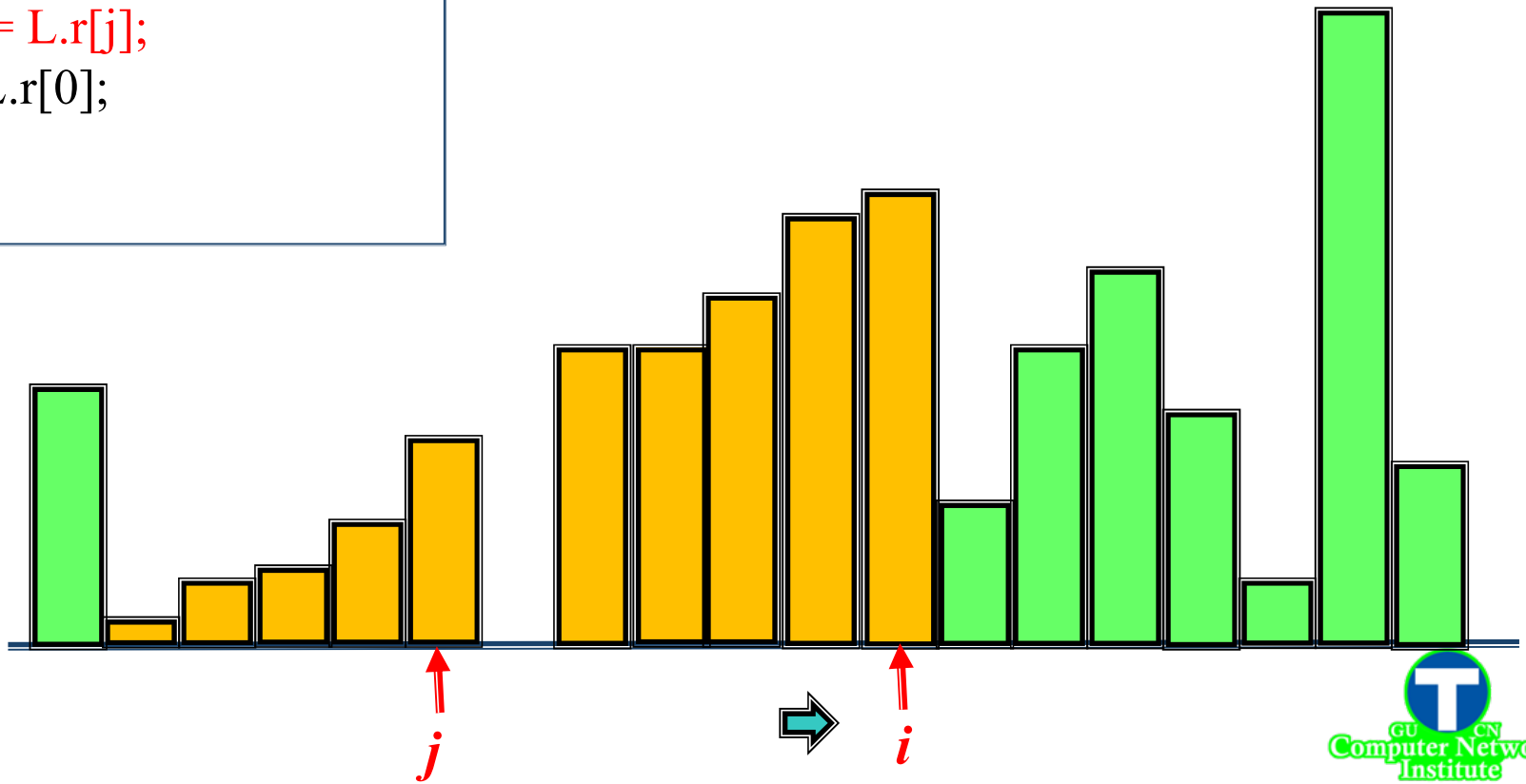


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

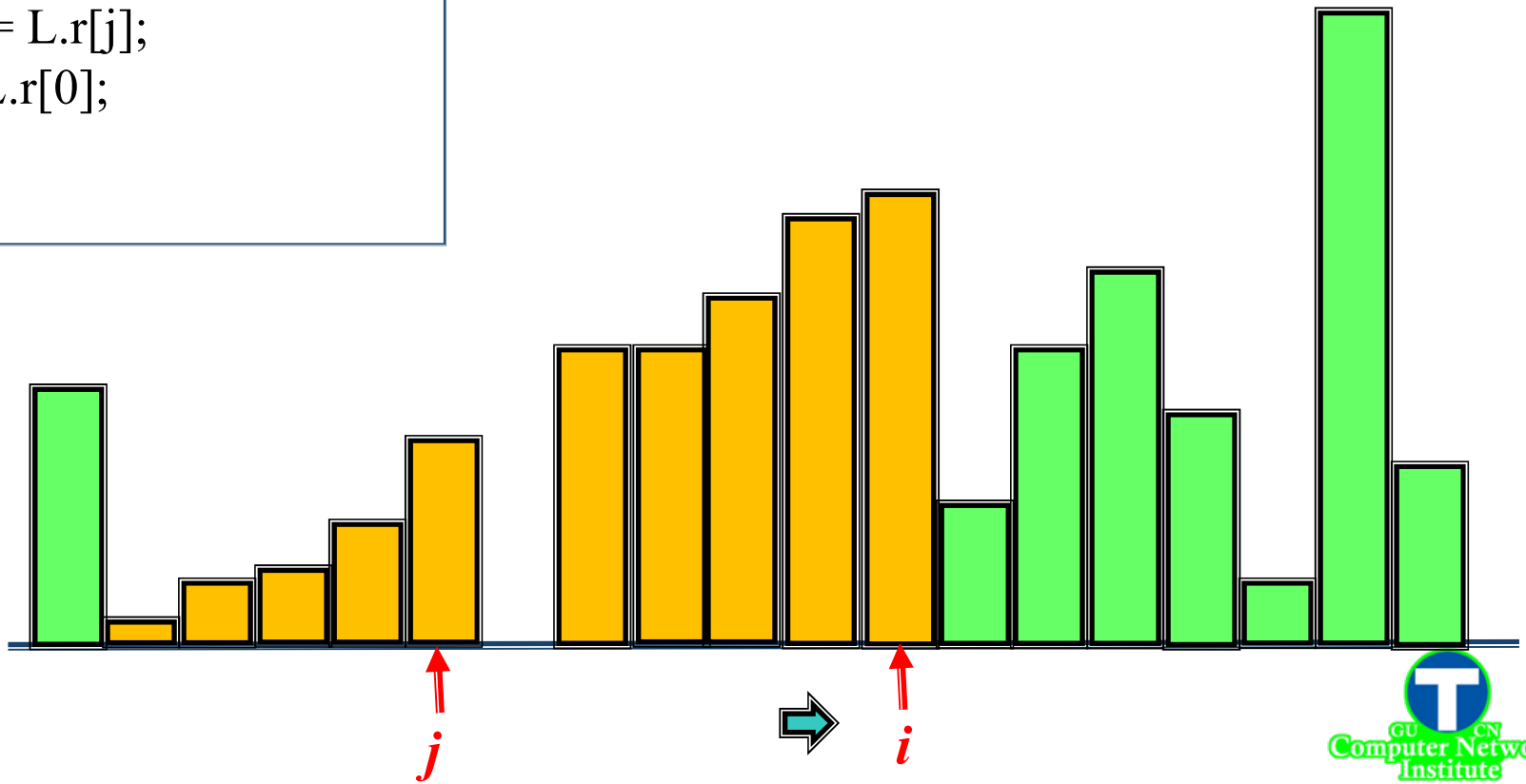


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

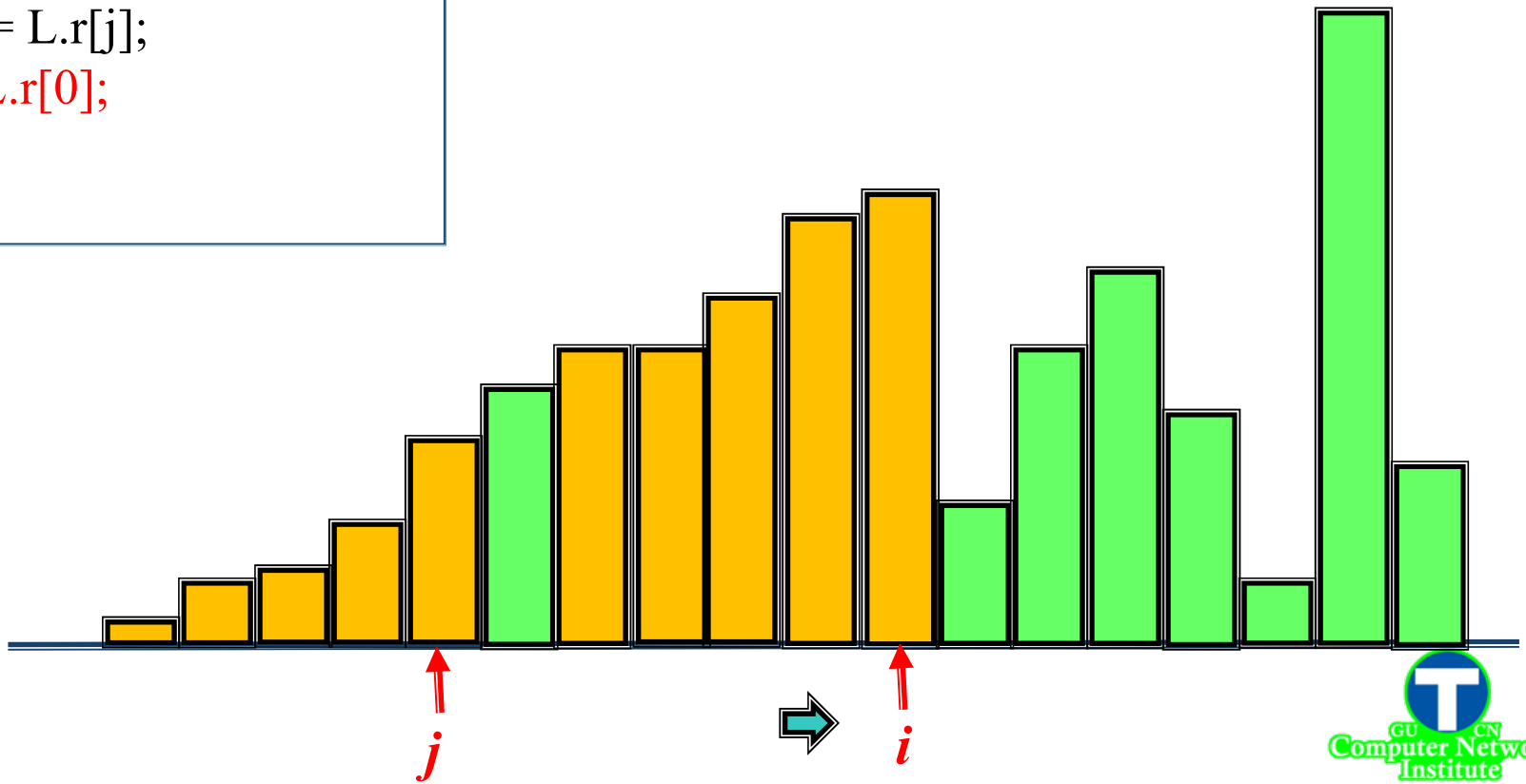


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。

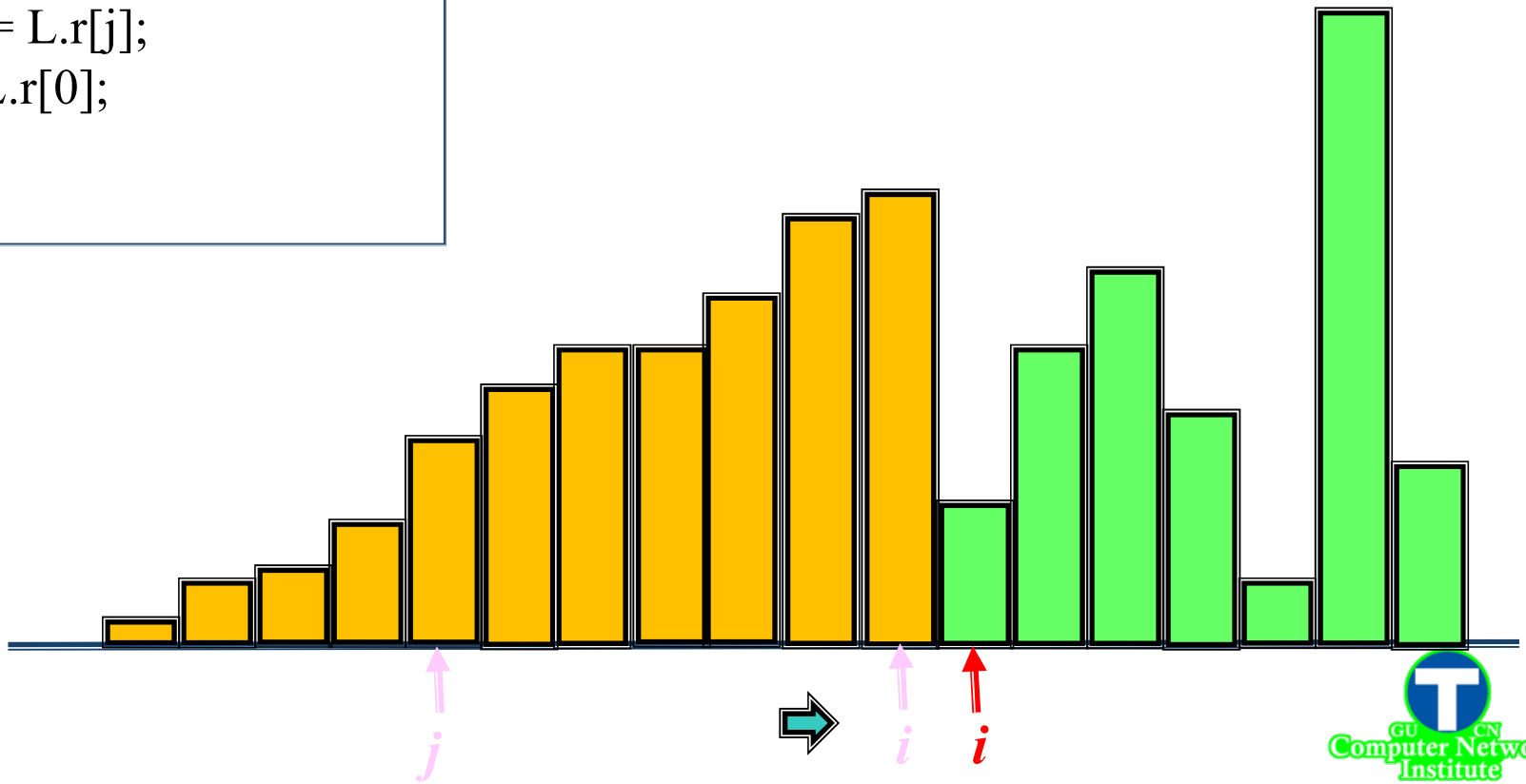


GU CN
Computer Networks
Institute

2 插入排序

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

直接插入排序：从第2个记录开始，逐个将每个元素插入到元素前面的有序子序列中去。



GU CN
Computer Networks
Institute

插入排序：计算复杂度

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

GU CN
Computer Networks
Institute

插入排序：计算复杂度

- 若待排序记录已按关键字从小到大排列(正序)关键字比较次数：

$$\sum_{i=2}^{n} 1 = n - 1$$

记录移动次数： 0

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

GU CN Computer Networks Institute

插入排序：计算复杂度

- 若待排序记录已按关键字从小到大排列(正序)关键字比较次数：

$$\sum_{i=2}^{n} 1 = n - 1$$

记录移动次数： 0

- 若待排序记录已按关键字从大到小排列(逆序)关键字比较次数：

$$\sum_{i=2}^{n} i = \frac{(n+2)(n-1)}{2}$$

记录移动次数：$\sum_{i=2}^{n}(i+1) = \frac{(n+4)(n-1)}{2}$

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

GU CN Computer Networks Institute

插入排序：计算复杂度

- 若待排序记录已按关键字从小到大排列(正序)关键字比较次数：

$$\sum_{i=2}^{n} 1 = n - 1$$

记录移动次数： 0

- 若待排序记录已按关键字从大到小排列(逆序)关键字比较次数：

$$\sum_{i=2}^{n} i = \frac{(n+2)(n-1)}{2}$$

记录移动次数：$\sum_{i=2}^{n}(i+1) = \frac{(n+4)(n-1)}{2}$

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

时间复杂度 $T(n) = O(n^2)$（最坏情况）

GU CN Computer Networks Institute

插入排序：计算复杂度

- 若待排序记录已按关键字从小到大排列(正序)关键字比较次数：

$$\sum_{i=2}^{n} 1 = n - 1$$

记录移动次数： 0

- 若待排序记录已按关键字从大到小排列(逆序)关键字比较次数：

$$\sum_{i=2}^{n} i = \frac{(n+2)(n-1)}{2}$$

记录移动次数：$\sum_{i=2}^{n}(i+1) = \frac{(n+4)(n-1)}{2}$

```
void InsertSort ( SqList &L)
{// 对顺序表L作直接插入排序
  for ( i=2; i<=L.length; ++i )
    if ( L.r[i-1].key > L.r[i].key ) {
        L.r[0] = L.r[i]; L.r[i]=L.r[i-1];
        for ( j=i-2;
                L.r[0].key < L.r[j].key; --j )
            L.r[j+1] = L.r[j];
        L.r[j+1] = L.r[0];
    }
} // InsertSort
```

时间复杂度 $T(n) = O(n^2)$（最坏情况）

空间复杂度 $S(n)=O(1)$

GU CN Computer Networks Institute

希尔排序（缩小增量排序）

- 取一个增量序列：$d_t, d_{t-1}, ..., d_1=1$。
- 对每一增量数d_k，把所有相隔d_k的元素做一组，组内进行直接插入排序。

13	27	48	55	4	49	38	65	97	76

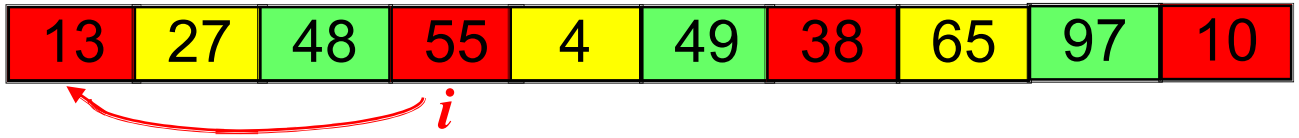
取$d_2=3$分组

13	27	48	55	4	49	38	65	97	76

GU CN Computer Networks Institute

希尔排序（缩小增量排序）

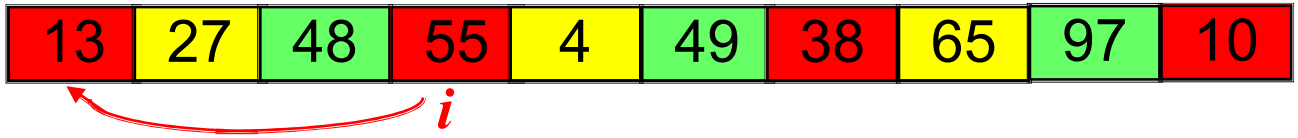
取d_2=3分组

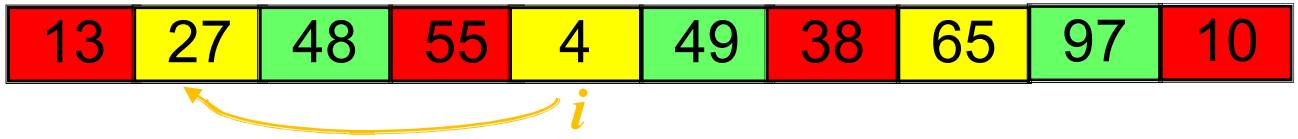


GU CN
Computer Networks
Institute

希尔排序（缩小增量排序）

取d_2=3分组

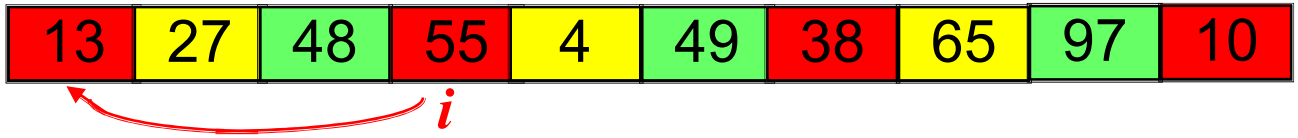


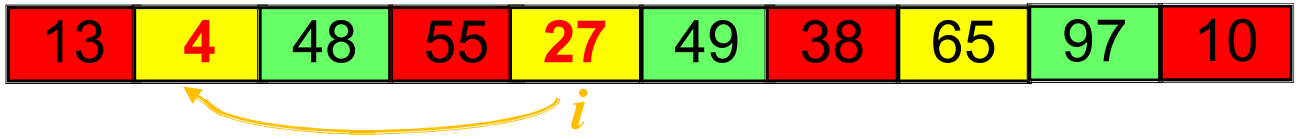


GU CN
Computer Networks
Institute

希尔排序（缩小增量排序）

取d_2=3分组

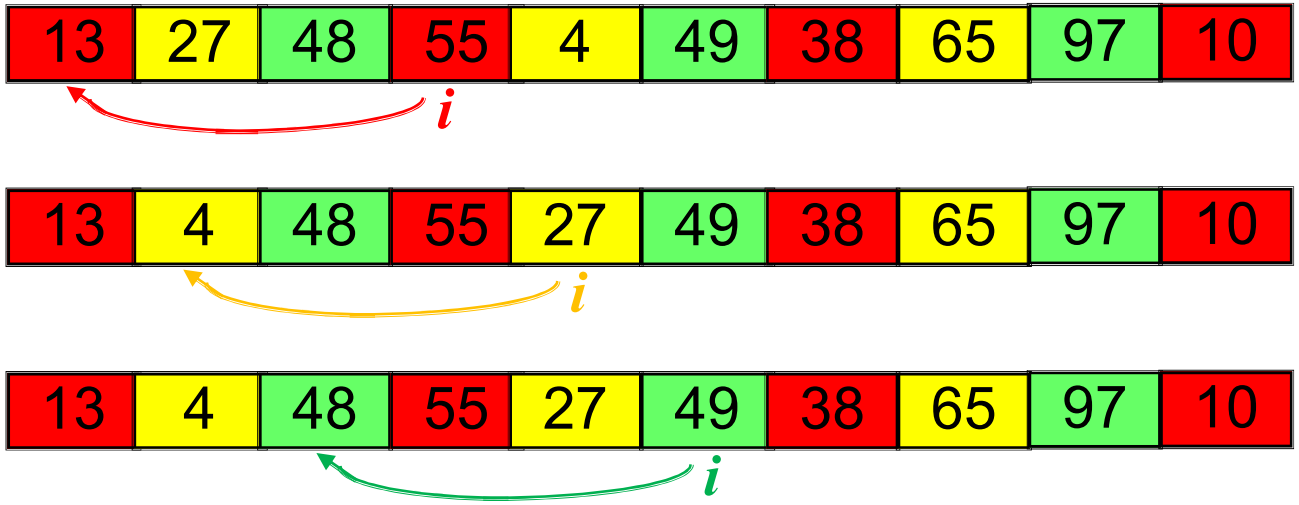




GU CN
Computer Networks
Institute

希尔排序（缩小增量排序）

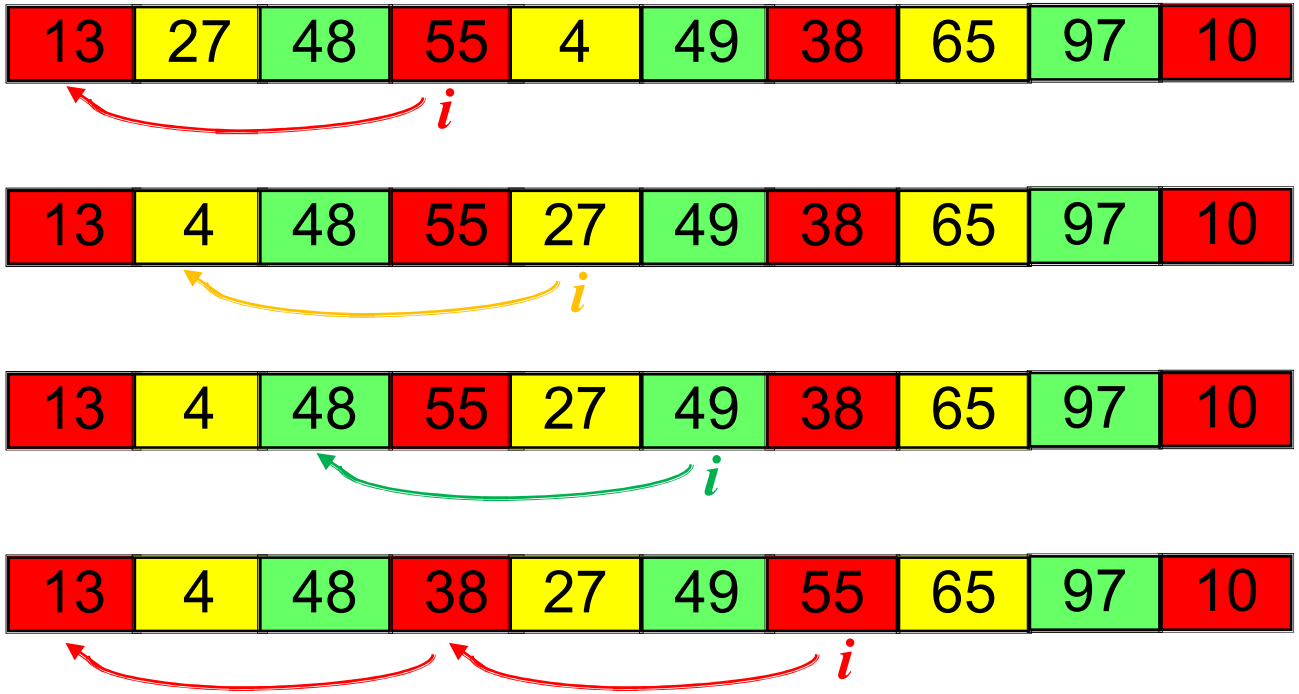
取$d_2=3$分组



GU CN
Computer Networks
Institute

希尔排序（缩小增量排序）

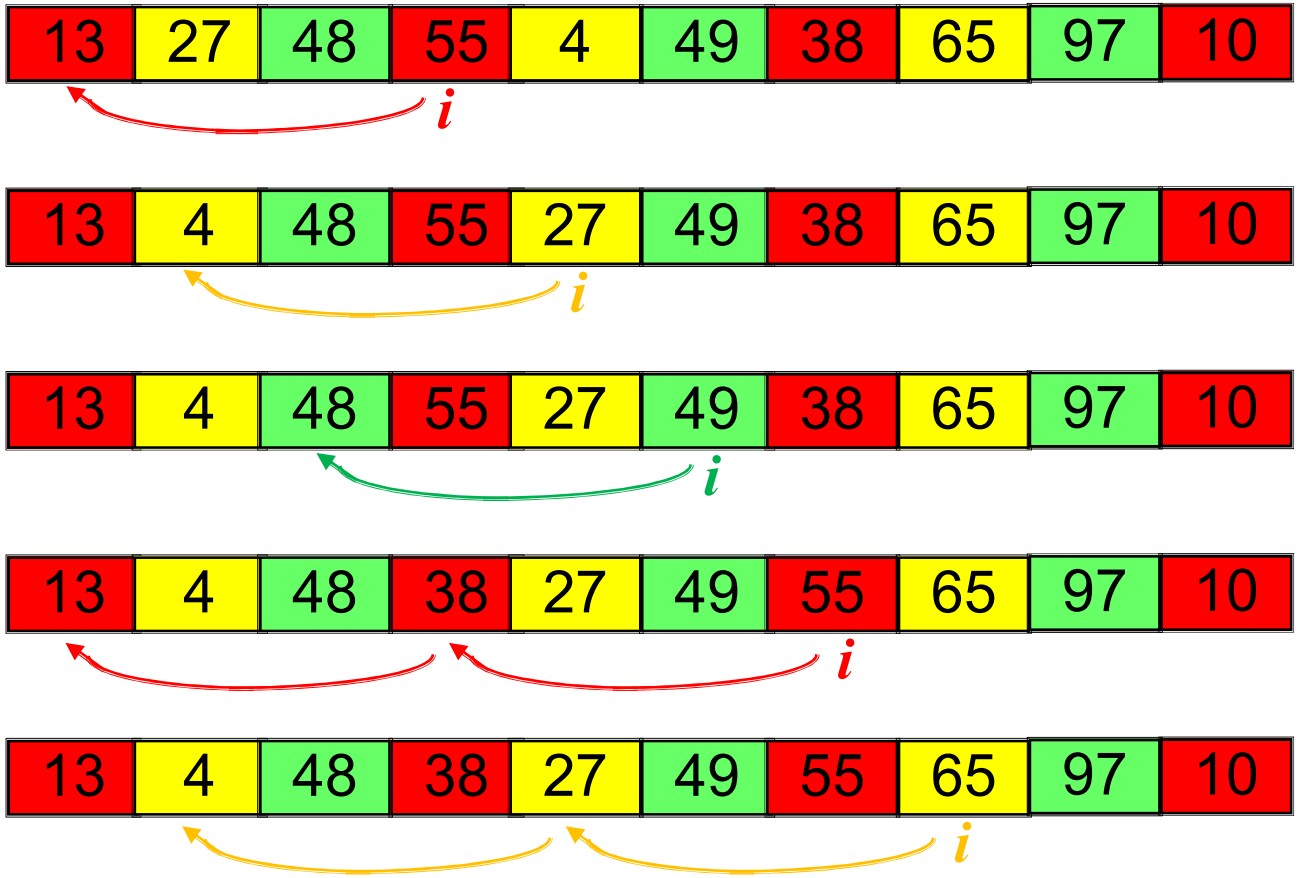
取$d_2=3$分组



GU CN
Computer Networks
Institute

希尔排序（缩小增量排序）

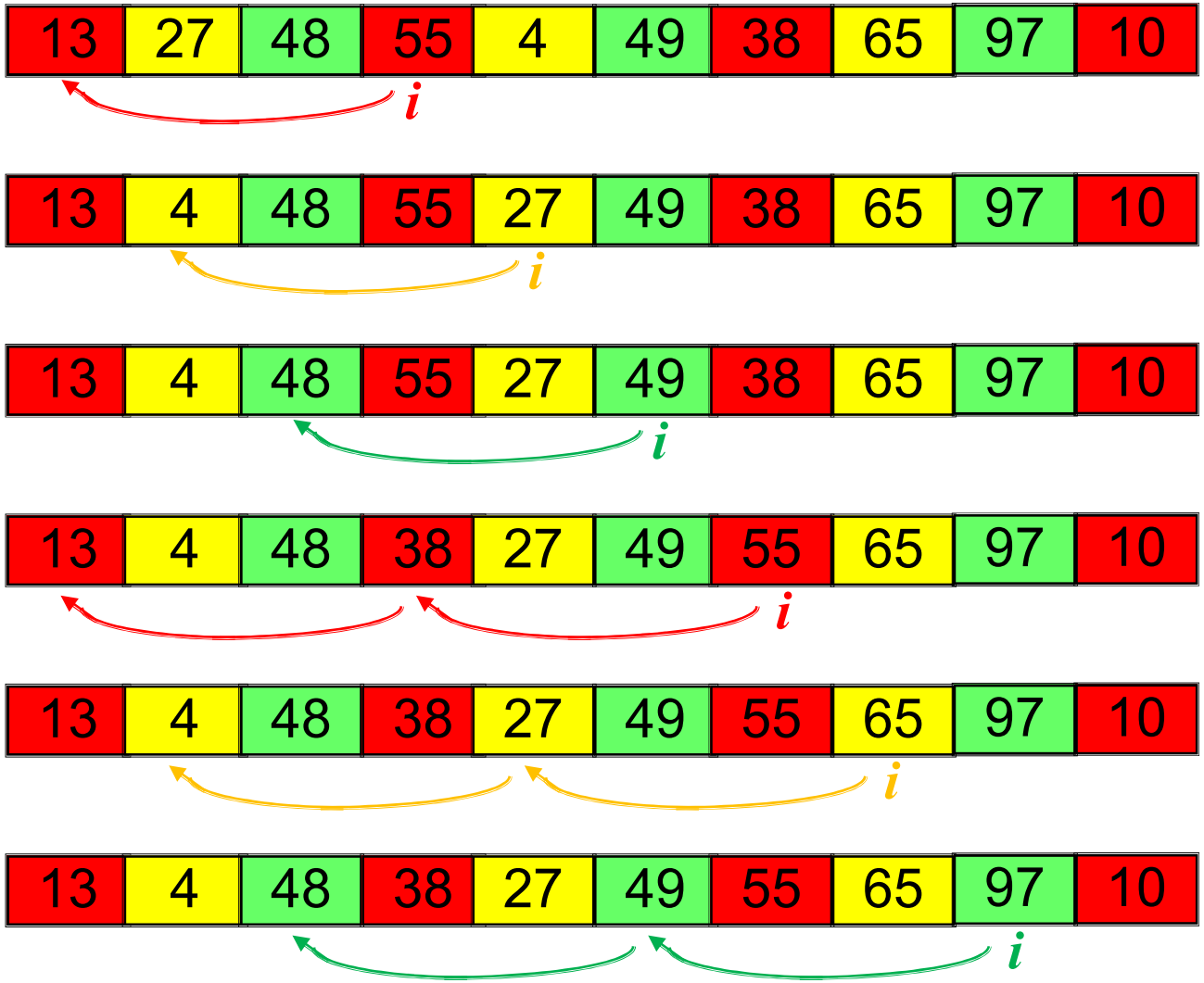
取$d_2=3$分组



GU CN Computer Networks Institute

希尔排序（缩小增量排序）

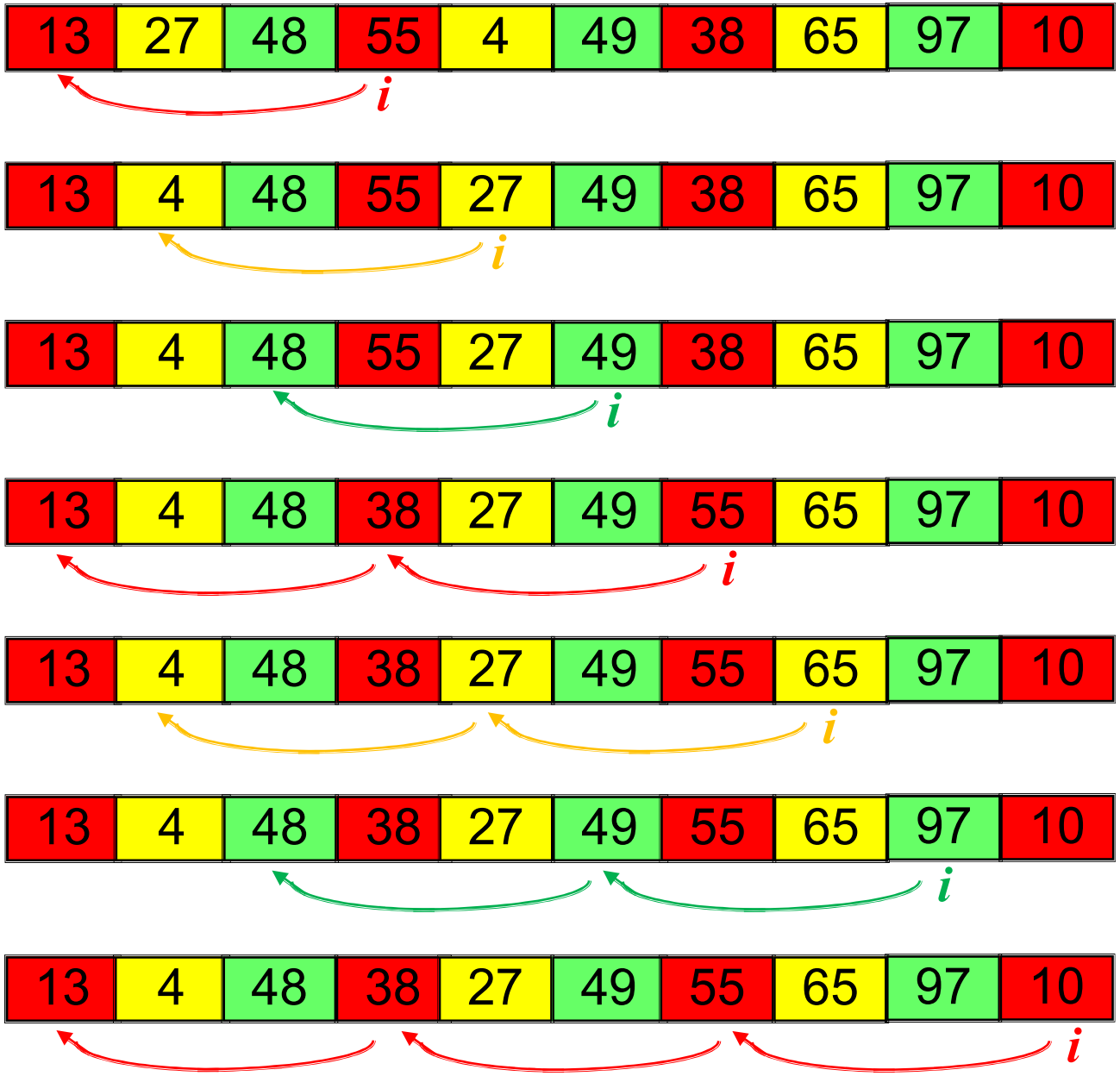
取$d_2=3$分组



GU CN Computer Networks Institute

希尔排序（缩小增量排序）

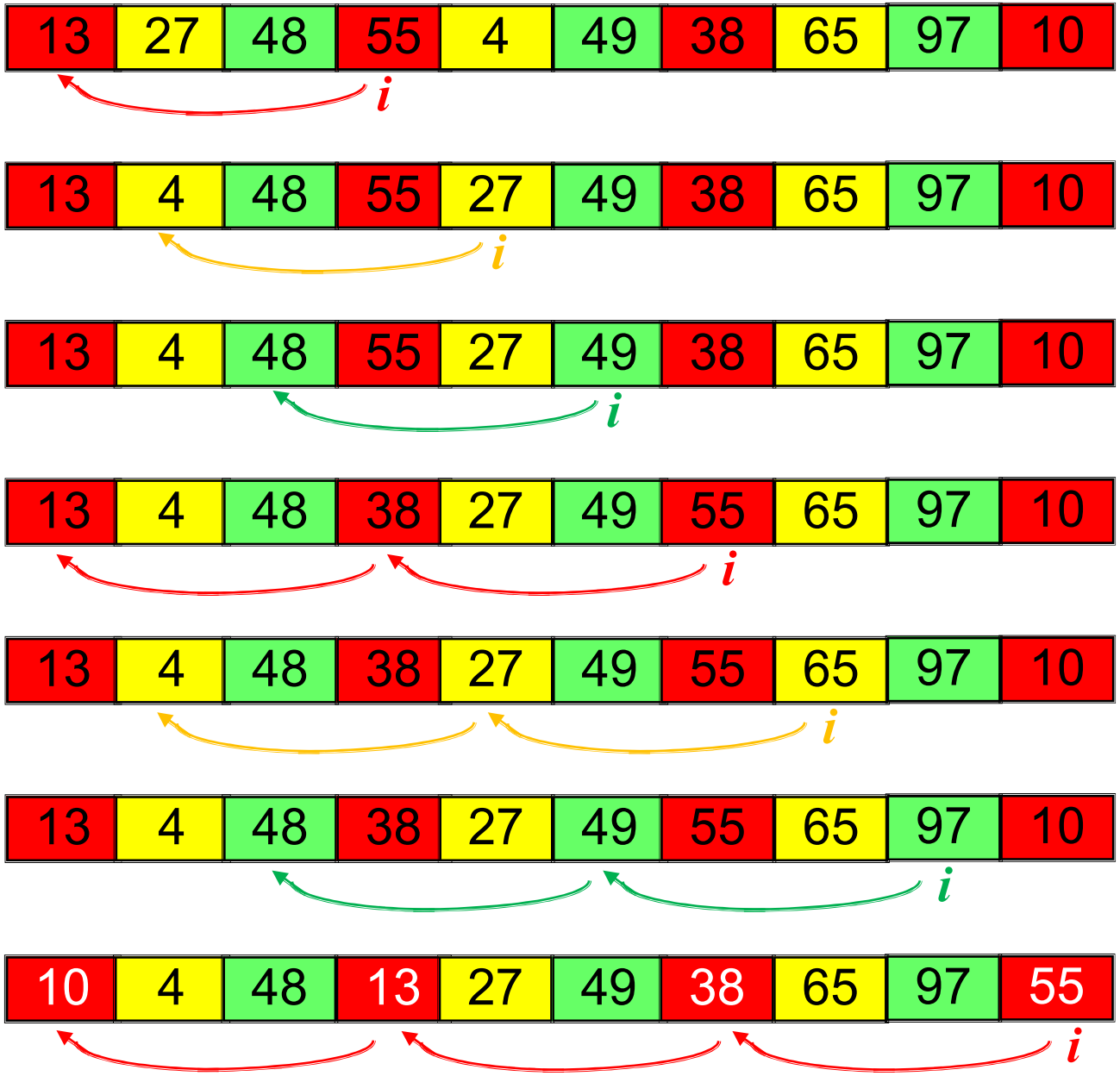
取d_2=3分组



GU CN Computer Networks Institute

希尔排序（缩小增量排序）

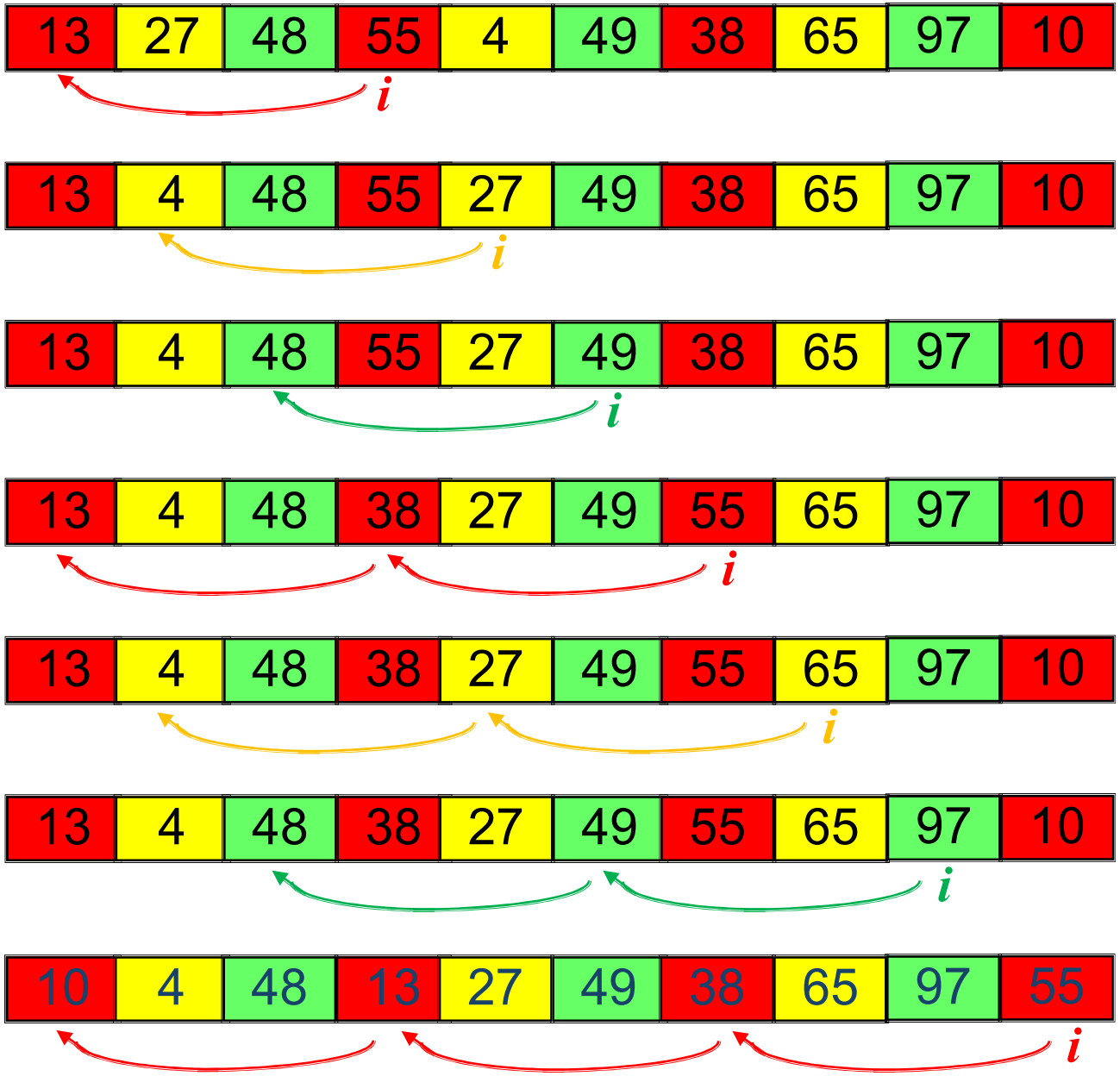
取d_2=3分组



GU CN Computer Networks Institute

希尔排序（缩小增量排序）

取d_2=3分组



GU CN Computer Networks Institute

希尔排序算法

```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
          L.r[0] = L.r[i];
          for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
              L.r[j+dk] = L.r[j];
           L.r[j+dk] = L.r[0];
} // if
} // ShellInsert
```

GU CN
Computer Networks
Institute

希尔排序算法

```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
          L.r[0] = L.r[i];
          for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
              L.r[j+dk] = L.r[j];
           L.r[j+dk] = L.r[0];
} // if
} // ShellInsert
```

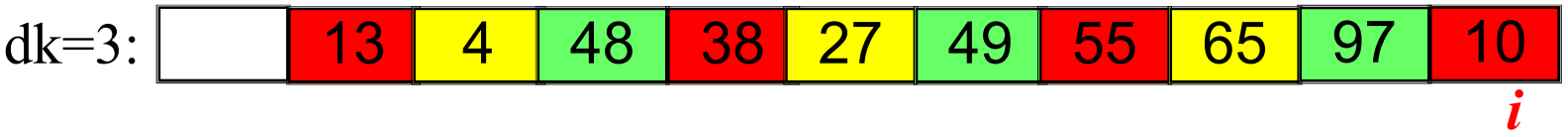
dk=3:

	13	4	48	38	27	49	55	65	97	10

GU CN Computer Networks Institute

希尔排序算法

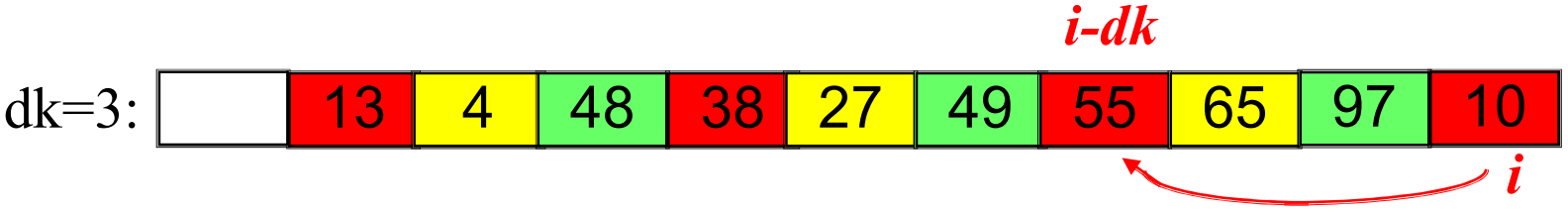
```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
          L.r[0] = L.r[i];
          for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
              L.r[j+dk] = L.r[j];
           L.r[j+dk] = L.r[0];
} // if
} // ShellInsert
```



GU CN
Computer Networks
Institute

希尔排序算法

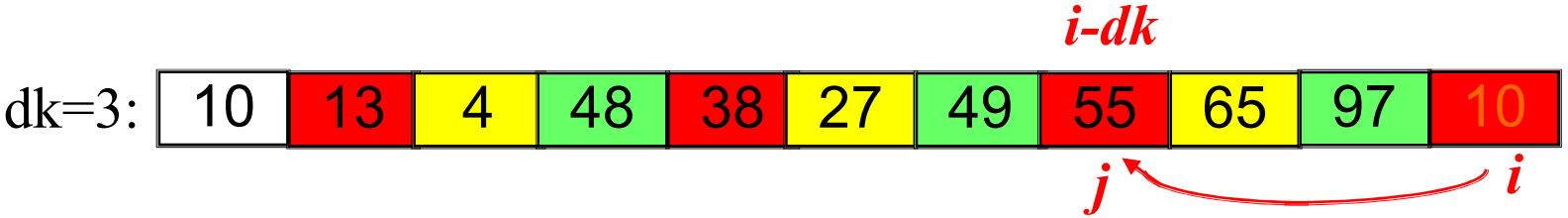
```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
          L.r[0] = L.r[i];
          for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
              L.r[j+dk] = L.r[j];
           L.r[j+dk] = L.r[0];
} // if
} // ShellInsert
```



GU CN
Computer Networks
Institute

希尔排序算法

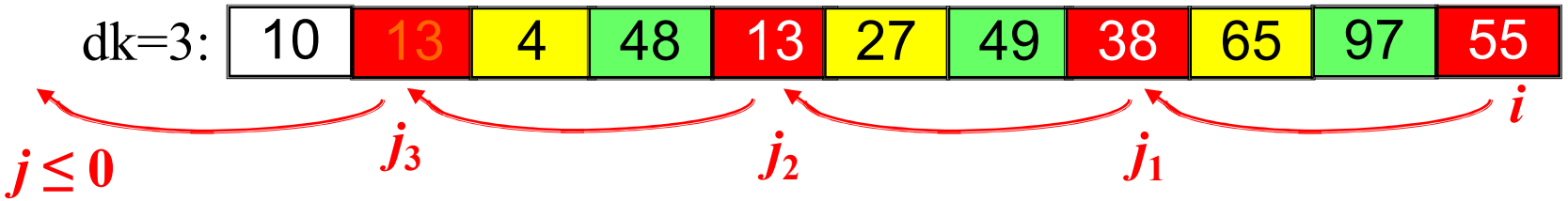
```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
         L.r[0] = L.r[i];
         for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
            L.r[j+dk] = L.r[j];
          L.r[j+dk] = L.r[0];
} // if
} // ShellInsert
```



GU CN
Computer Networks
Institute

希尔排序算法

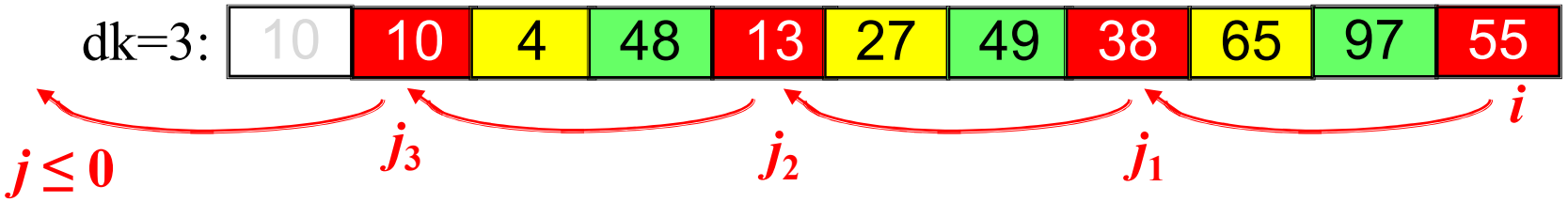
```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
          L.r[0] = L.r[i];
          for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
              L.r[j+dk] = L.r[j];
          L.r[j+dk] = L.r[0];
} // if
} // ShellInsert
```



GU CN
Computer Networks
Institute

希尔排序算法

```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
          L.r[0] = L.r[i];
          for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
              L.r[j+dk] = L.r[j];
          L.r[j+dk] = L.r[0];
} // if
} // ShellInsert
```



GU CN
Computer Networks
Institute

希尔排序算法

```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
          L.r[0] = L.r[i];
          for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
              L.r[j+dk] = L.r[j];
          L.r[j+dk] = L.r[0];
} // if
} // ShellInsert
```

dk=3:	10	10	4	48	13	27	49	38	65	97	55

希尔排序算法

```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
         L.r[0] = L.r[i];
         for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
            L.r[j+dk] = L.r[j];
          L.r[j+dk] = L.r[0];
} // if
} // ShellInsert


void ShellSort (SqList &L, int dlta[], int t)
{
       for (k=0; k<t; ++k)
          ShellInsert(L, dlta[k]);
} // ShellSort
```

GU CN Computer Networks Institute

希尔排序算法

```
void ShellInsert (SqList &L, int dk)
{   // 对顺序表L作一趟增量为dk的希尔排序
    for ( i=dk+1; i<=L.length; ++i )
      if ( L.r[i].key < L.r[i-dk].key ) {
         L.r[0] = L.r[i];
         for ( j=i-dk; j>0 && L.r[0].key < L.r[j].key; j=j-dk )
            L.r[j+dk] = L.r[j];
          L.r[j+dk] = L.r[0];
} // if
} // ShellInsert
```

```
void ShellSort (SqList &L, int dlta[], int t)
{
       for (k=0; k<t; ++k)
          ShellInsert(L, dlta[k]);
} // ShellSort
```

dlta

d1	d2	d3	d4

GU CN Computer Networks Institute

希尔排序算法讨论

GU CN
Computer Networks
Institute

希尔排序算法讨论

- 子序列的构成不是简单“逐段分割”，而是将相隔某个增量的元素组成组。

GU CN
Computer Networks
Institute

希尔排序算法讨论

- 子序列的构成不是简单“逐段分割”，而是将相隔某个增量的元素组成组。
- 希尔排序可提高排序速度：分组后每组的n值减小，而n值小时插入排序效率高($T(n)=O(n^2)$)。

GU CN
Computer Networks
Institute

希尔排序算法讨论

- 子序列的构成不是简单“逐段分割”，而是将相隔某个增量的元素组成组。
- 希尔排序可提高排序速度：分组后每组的n值减小，而n值小时插入排序效率高($T(n)=O(n^2)$)。
- 最后一趟增量为1时，序列已“基本有序”，故插入排序所做比较和移动操作都很少。

GU CN
Computer Networks
Institute

希尔排序算法讨论

- 子序列的构成不是简单“逐段分割”，而是将相隔某个增量的元素组成组。
- 希尔排序可提高排序速度：分组后每组的n值减小，而n值小时插入排序效率高($T(n)=O(n^2)$)。
- 最后一趟增量为1时，序列已“基本有序”，故插入排序所做比较和移动操作都很少。
- 希尔排序的时间复杂度和所取增量序列相关。已有学者证明，当增量序列为 $dk = 2^{t-k-1}$ $(k= 0, 1, \ldots, t-1)$ 时，希尔排序的时间复杂度为$O(n^{3/2})$。

GU CN Computer Networks Institute

希尔排序算法讨论

- 子序列的构成不是简单“逐段分割”，而是将相隔某个增量的元素组成组。
- 希尔排序可提高排序速度：分组后每组的n值减小，而n值小时插入排序效率高($T(n)=O(n^2)$)。
- 最后一趟增量为1时，序列已“基本有序”，故插入排序所做比较和移动操作都很少。
- 希尔排序的时间复杂度和所取增量序列相关。已有学者证明，当增量序列为 $dk = 2^{t-k-1}$ $(k= 0, 1, \ldots, t-1)$ 时，希尔排序的时间复杂度为$O(n^{3/2})$。
- 增量序列的取法
  (1) 无除1以外的公因子
  (2) 采用缩小增量法，最后一个增量值必须为1

GU CN Computer Networks Institute

第10章 测验题

GU CN
Computer Networks
Institute

单选题 1分

设置

对一组数据（2,12,16,88,5,10）进行排序，若前三趟排序结果如下，则采用的排序方法可能是（ ）。

第一趟排序结果：2,12,16,5,10,88

第二趟排序结果：2,12,5,10,16,88

第三趟排序结果：2,5,10,12,16,88

A 冒泡排序

B 希尔排序

C 归并排序

D 基数排序

提交

GU CN Computer Networks Institute

单选题 1分

设置

在堆排序中，堆的形状是一棵（ ）。

A 二叉排序树

B 满二叉树

C 完全二叉树

D 平衡二叉树

提交

GU CN Computer Networks Institute

填空题 1分

设置

快速排序在 [填空1] 情况下最易发挥其长处。

作答

GU CN Computer Networks Institute

填空题 0分

设置

若一组记录的排序码为（46,79,56,38,40,84），利用堆排序建立的初始堆是 [填空1] 。

作答

GU CN Computer Networks Institute