C语言程序设计

计算机科学与技术学院

© 东华大学

数组

- 一维数组
- 二维数组
- 字符数组
- 数组相关算法

© 东华大学

数组相关常见算法

- 一维数组相关算法
  - 查找(顺序、二分)
  - 计算最大值/位置/和/平均值
  - 倒置
  - 排序（选择、冒泡）
  - 插入
  - 删除

- 二维数组相关算法
  - 转置
  - 计算最大值
  - 各行各列的最大值/求和
  - 杨辉三角形
  - 上三角元素之和

$$\mathbf{U} = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \cdots & u_{1,n} \\ & u_{2,2} & u_{2,3} & \cdots & u_{2,n} \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & u_{n-1,n} \\ 0 & & & & u_{n,n} \end{bmatrix}$$

© 东华大学

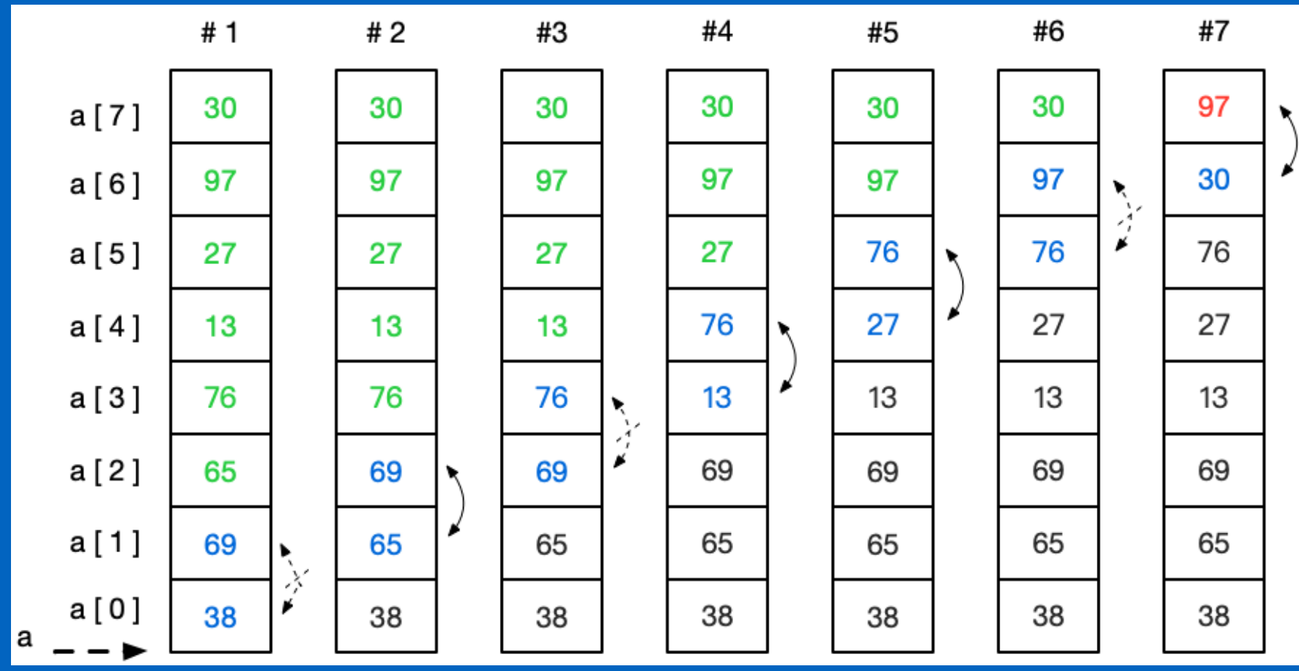
排序

- 排序(sort)是将一组无序的记录按某字段值的升序或降序形式重新组织，形成按此字段值有序的记录序列
- 常用算法
  - 冒泡排序
  - 选择排序

© 东华大学

冒泡排序(Bubble Sort)

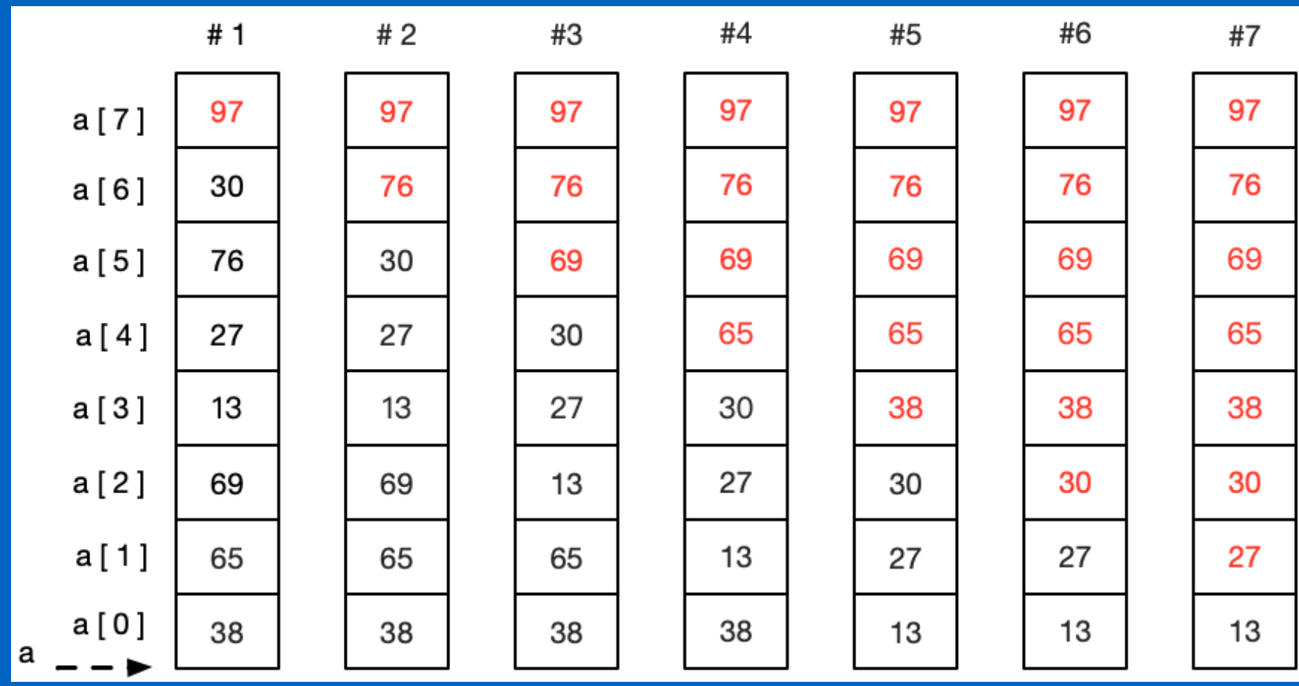
- 对 n 个数据 a[0] ~ a[n-1] 进行升序排序
- 一趟冒泡过程：
  - Step 1: 比较第一个数 a[0] 与第二个数 a[1]
  - 若为逆序，则交换这两个数；
  - Step 2: 比较第二个数 a[1] 与第三个数 a[2]]
  - 若为逆序，则交换这两个数；
  - … 依次类推
  - 直至第n-1个数 a[n-2] 和第n个数 a[n-1] 比较完毕为止
- 第一趟冒泡过程结果：
  - 最大值被交换至最后一个元素位置 a[n-1]



© 东华大学

冒泡排序

- 对前n-1个数 a[0] ~ a[n-2] 进行第二趟冒泡过程
  - 其中最大值被交换至第n-1个元素位置 a[n-2]
- 重复上述过程
  - 对剩余数据进行下一趟冒泡过程
    - 其中最大值依次被交换至在 a[n-2]、a[n-3]、...
- 经过 n-1 趟冒泡过程
  - 最后一次对 a[0] ~ a[1] 进行冒泡
    - 其中最大值被交换至第2个元素位置 a[1]
- 排序结束



	# 1	# 2	#3	#4	#5	#6	#7
a[7]	97	97	97	97	97	97	97
a[6]	30	76	76	76	76	76	76
a[5]	76	30	69	69	69	69	69
a[4]	27	27	30	65	65	65	65
a[3]	13	13	27	30	38	38	38
a[2]	69	69	13	27	30	30	30
a[1]	65	65	65	13	27	27	27
a[0]	38	38	38	38	13	13	13

a

© 东华大学

- n 个元素：需进行 n-1 趟冒泡（外循环）

```
for( i = 0; i < n-1; i++ ) {
    第 i 趟冒泡;
}
```

- 第？ 趟冒泡规律

第？ 趟	范围	比较次数
0	a[0] ~ a[n-1]	n-1
1	a[0] ~ a[n-2]	n-2
……	……	……
i	a[0] ~ a[n-1-i]	n-1-i
……	……	……
n-2	a[0] ~ a[1]	1

- 第 i 趟冒泡（内循环）

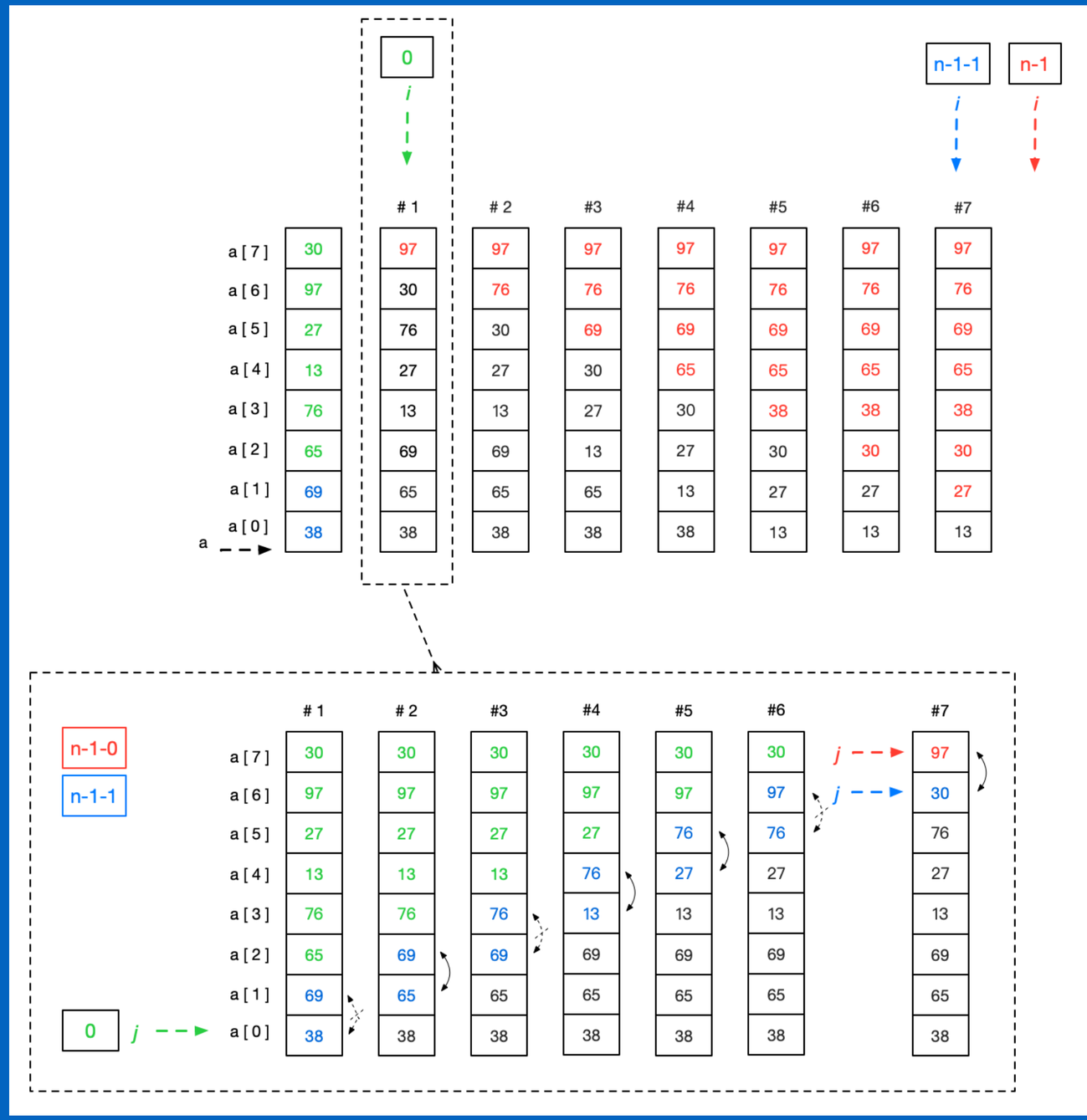
```
// j:  a[0] ~ a[n-1-i-1]
for( j = 0; j < n-1-i; j++) {
    //如果当前数比后数大
    if ( a[j] > a[j+1] ) {
        //交换两个数据;
        a[j] <-> a[j+1];
    }
}
```

	# 1	# 2	#3	#4	#5	#6	#7
a[7]	97	97	97	97	97	97	97
a[6]	30	76	76	76	76	76	76
a[5]	76	30	69	69	69	69	69
a[4]	27	27	30	65	65	65	65
a[3]	13	13	27	30	38	38	38
a[2]	69	69	13	27	30	30	30
a[1]	65	65	65	13	27	27	27
a[0]	38	38	38	38	13	13	13

a ▶

© 东华大学

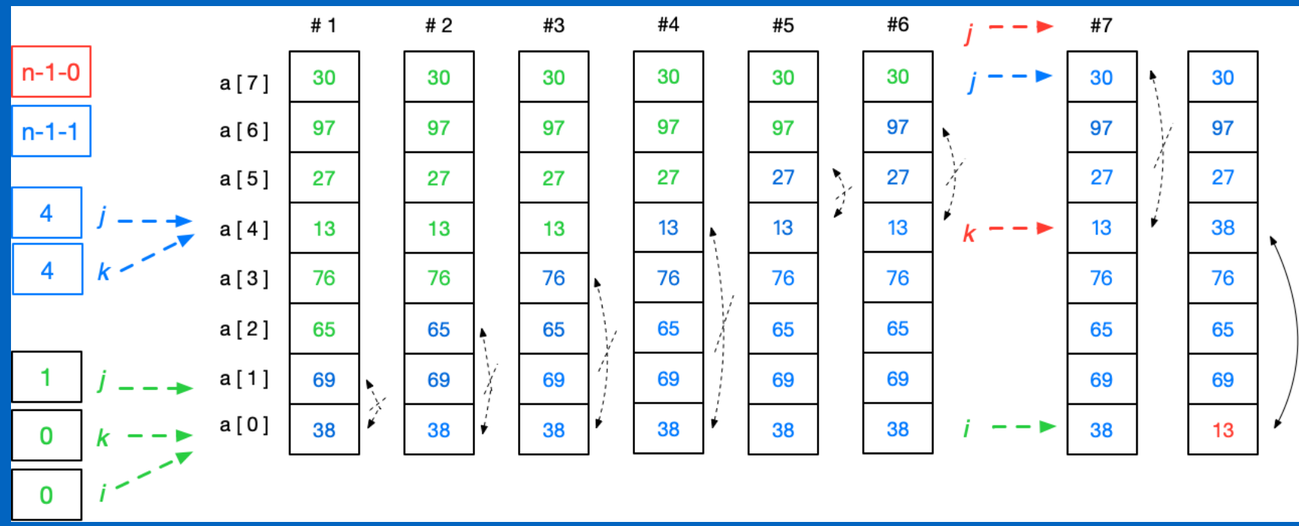
```c
#include <stdio.h>
#define SIZE 100
int main (){
    int a[SIZE],i,j,t,n;
    printf("Enter the number of integers: ");
    scanf("%d",&n);

    printf("Enter the elements of array:");
    for(i = 0; i < n; i++)
        scanf("%d",&a[i]);
    printf("\n");

    for(i = 0; i < n-1; i++)
        for(j = 0; j < n-1-i; j++)
            if(a[j] > a[j+1]){
                t = a[j]; a[j] = a[j+1]; a[j+1] = t;
            }

    printf("The sorted array:");
    for(i = 0; i < n; i++)
        printf("%d ",a[i]);

    return 0;
}
```



© 东华大学

选择排序

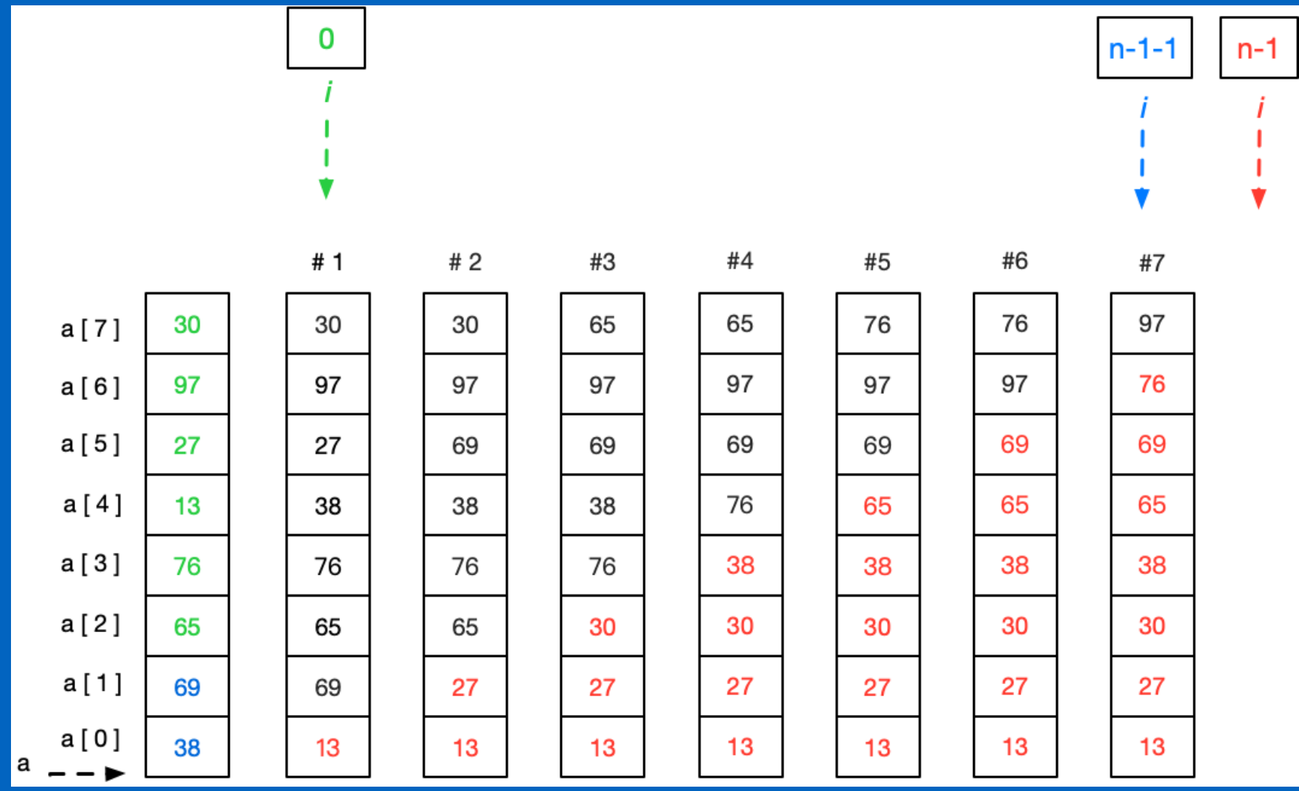
- 对 n 个数据 a[0] ~ a[n-1] 进行升序排序
- 一趟选择过程：
  - Step 1: 默认 其中 a[0]为最小值（位置）k = 0；
  - Step 2: 最小值 a[k] 与第二个数 a[1] 比较
  - 若为逆序，则更新最小值（位置）k = 1；
  - … 依次类推
  - 直至 a[k]与第n个数 a[n-1] 比较完毕
- 第一趟选择过程结果
  - 找到其中最小值（位置）a[k];
  - 如果 k != 0，把 a[k] 交换至第一个元素位置 a[0]



© 东华大学

选择排序

- 对剩余的 n-1个数 a[1] ~ a[n-1] 进行第二趟选择过程
  - 找到其中最小值（位置）a[k];
  - 如果 k != 1，把 a[k] 交换至第二个元素位置 a[1]；
- 重复上述过程
  - 对剩余数据进行下一趟选择过程
  - 找到其中最小值（位置）a[k];
  - 如果 k != 2、k != 3，把a[k] 交换至 a[2]、a[3]、...
- 经过 n-1 趟选择过程
  - 最后一次对 a[n-2] ~ a[n-1] 进行选择
  - 找到其中最小值（位置）a[k];
  - 如果k != n-2, 把a[k] 交换至倒数第二个元素位置 a[n-2]
- 排序结束



© 东华大学

• n 个元素，进行 n-1 趟选择（外循环）

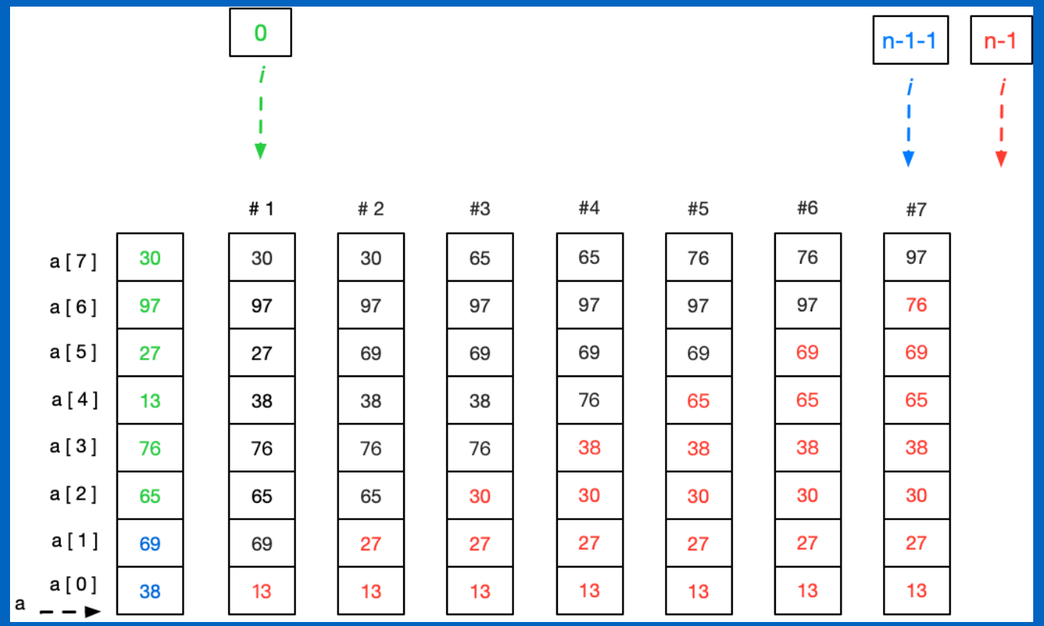
```
for( i = 0; i < n-1; i++ ){
    第 i 趟选择过程;
}
```

• 第 ? 趟选择规律

第 ? 趟	范围	比较次数
0	a[0] ~ a[n-1]	n-1
1	a[1] ~ a[n-1]	n-2
.....	.....	.....
i	a[i] ~ a[n-1]	n-1-i
.....	.....	.....
n-2	a[n-2] ~ a[n-1]	1

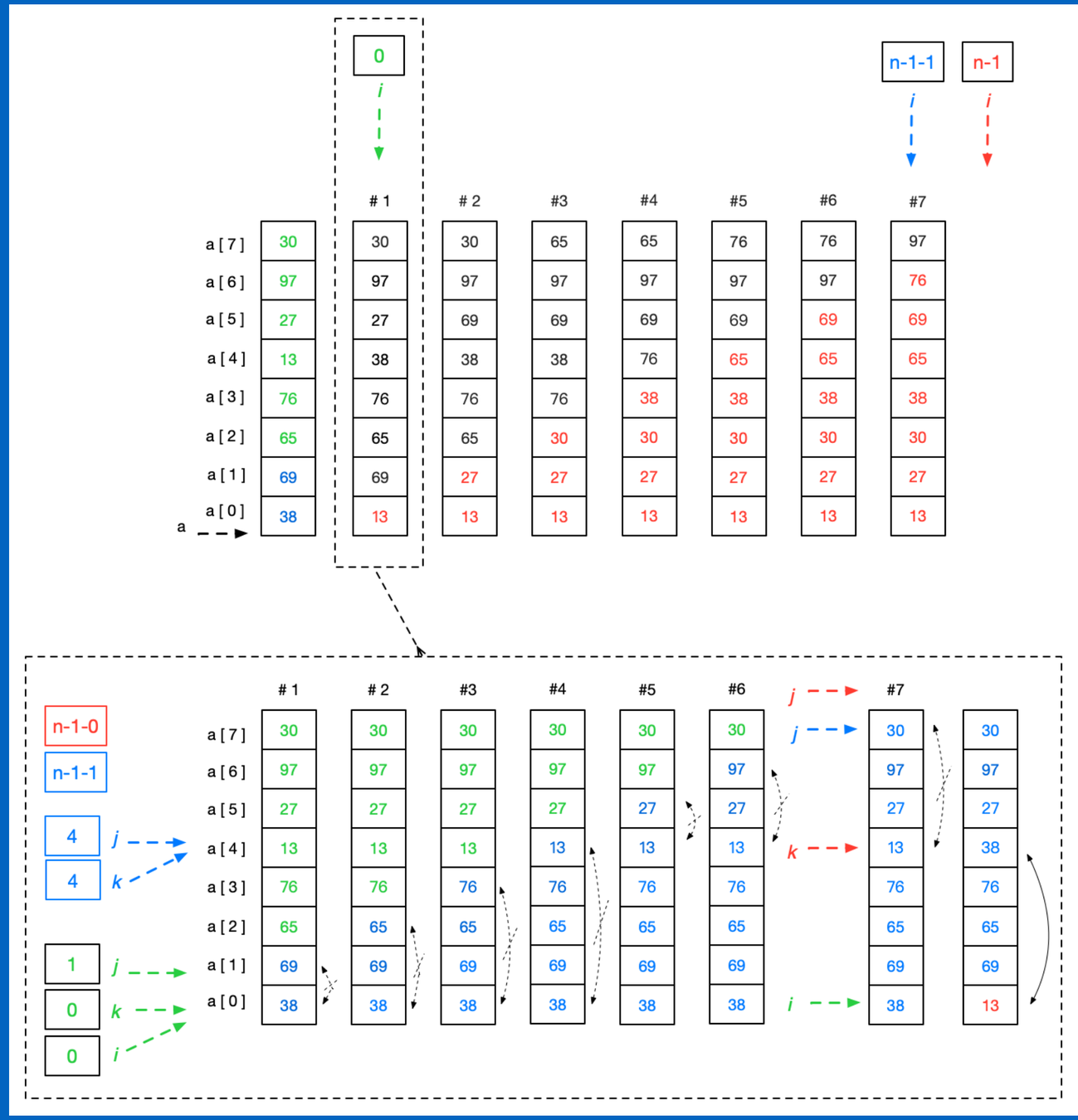
• 第 i 趟选择（内循环）

```
// a[k]: a[i] ~ a[n-1]中的最小值
k <- i; //默认 a[i]最小
// j : a[i+1] ~ a[n-1]
for( j = i + 1;  j < n;  j++) {
    if( a[j] < a[k] )  //如果a[j]更小
        k <- i; // k 就跟上去
}
// 如果a[k]的位置有变化
if( k! = i )
    // 把a[k]交换到a[i]
    a[i] <-> a[k];
```



© 东华大学

```c
#include <stdio.h>
#define SIZE 100
int main (){
    int a[SIZE],i,j,t,n,k;
    printf("Enter the number of integers: ");
    scanf("%d",&n);

    printf("Enter the elements of array:");
    for(i = 0; i < n; i++)
        scanf("%d",&a[i]);
    printf("\n");

    for(i = 0; i < n-1; i++){
        k = i;
        for( j = i + 1; j < n; j++)
            if(a[j] < a[k])
                k = j;
        if( k != i ) {
            t = a[i]; a[i] = a[k]; a[k] = t;
        }
  }

    printf("The sorted array:");
    for(i = 0; i < n; i++)
        printf("%d ",a[i]);

    return 0;
}
```



© 东华大学

排序算法

- 冒泡排序（bubble sort） — $O(n^2)$
- 选择排序 （selection sort） — $O(n^2)$
- 鸡尾酒排序 (Cocktail sort) — $O(n^2)$
- 插入排序 （insertion sort） — $O(n^2)$
- 桶排序 （bucket sort） — O(n);
- 计数排序 (counting sort) — O(n+k);
- 归并排序 （merge sort） — O(n log n);
- 原地归并排序 — $O(n^2)$
- 二叉树排序 （Binary tree sort） — O(n log n);
- 鸽巢排序 (Pigeonhole sort) — O(n+k);
- 基数排序 （radix sort） — O(n•k);
- Gnome sort — $O(n^2)$
- Library sort — O(n log n)
- Comb sort — O(n log n)
- 堆排序 （heapsort） — O(n log n)
- Smoothsort — O(n log n)
- 快速排序 （quicksort） — O(n log n)
- Introsort — O(n log n)
- Patience sorting — O(n log n + k)
- ...
- 内（存）排序、外（存）排序
- 并行排序、分布式排序：MapReduce
- 整数、文本、网页、图像、...

© 东华大学

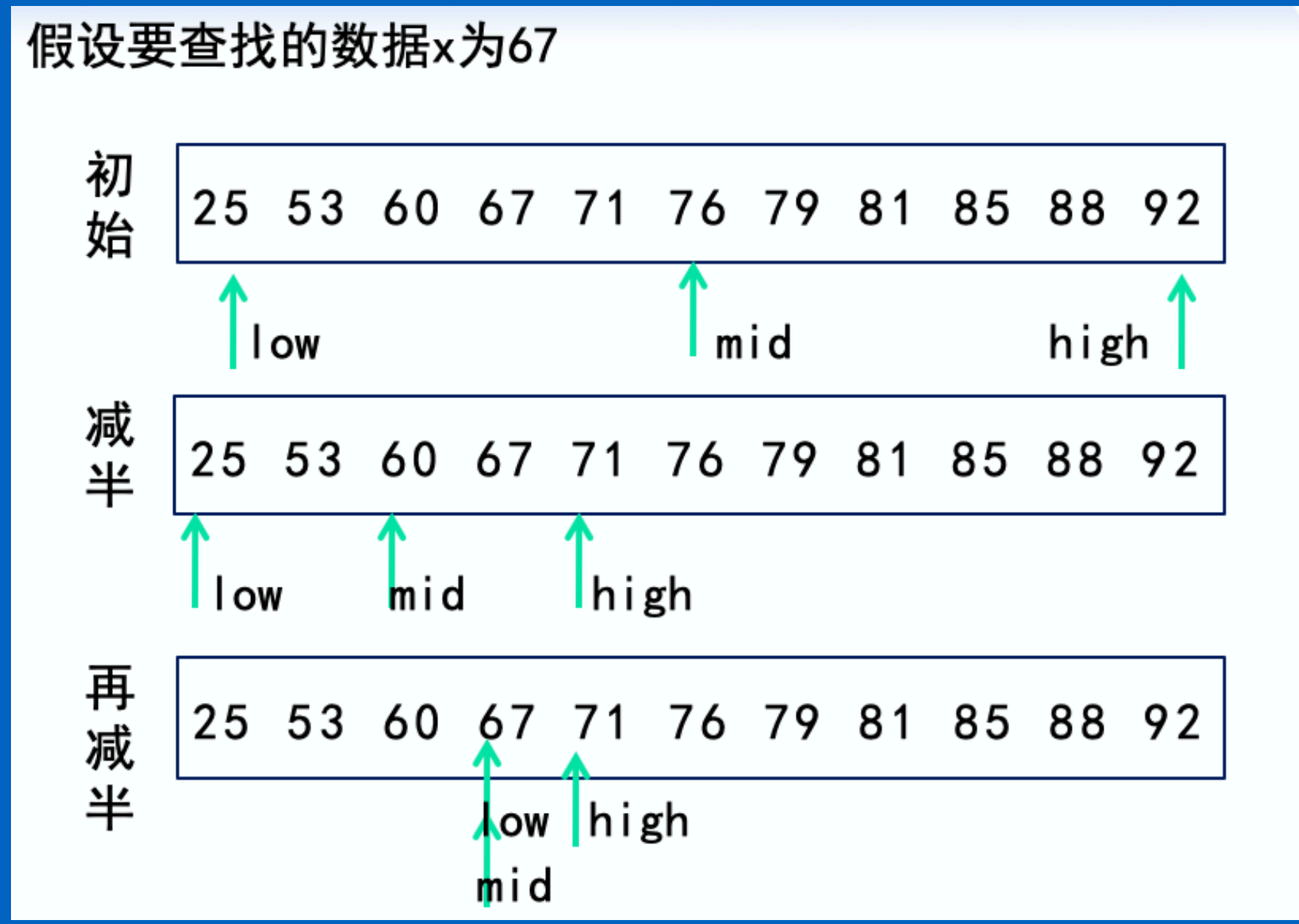
麻省理工学院(MIT)“开放式课程网页”

- Introduction to Algorithms, 算法导论
- https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/
- Course Features:
  - Video lectures
  - Captions/transcript
  - Lecture notes
  - Assignments: problem sets with solutions
  - Assignments: programming with examples
  - Exams and solutions
  - Recitation videos
- Course Description
  This course provides an introduction to mathematical modeling of computational problems. It covers the common algorithms, algorithmic paradigms, and data structures used to solve these problems. The course emphasizes the relationship between algorithms and programming, and introduces basic performance measures and analysis techniques for these problems.

© 东华大学

二分法查找

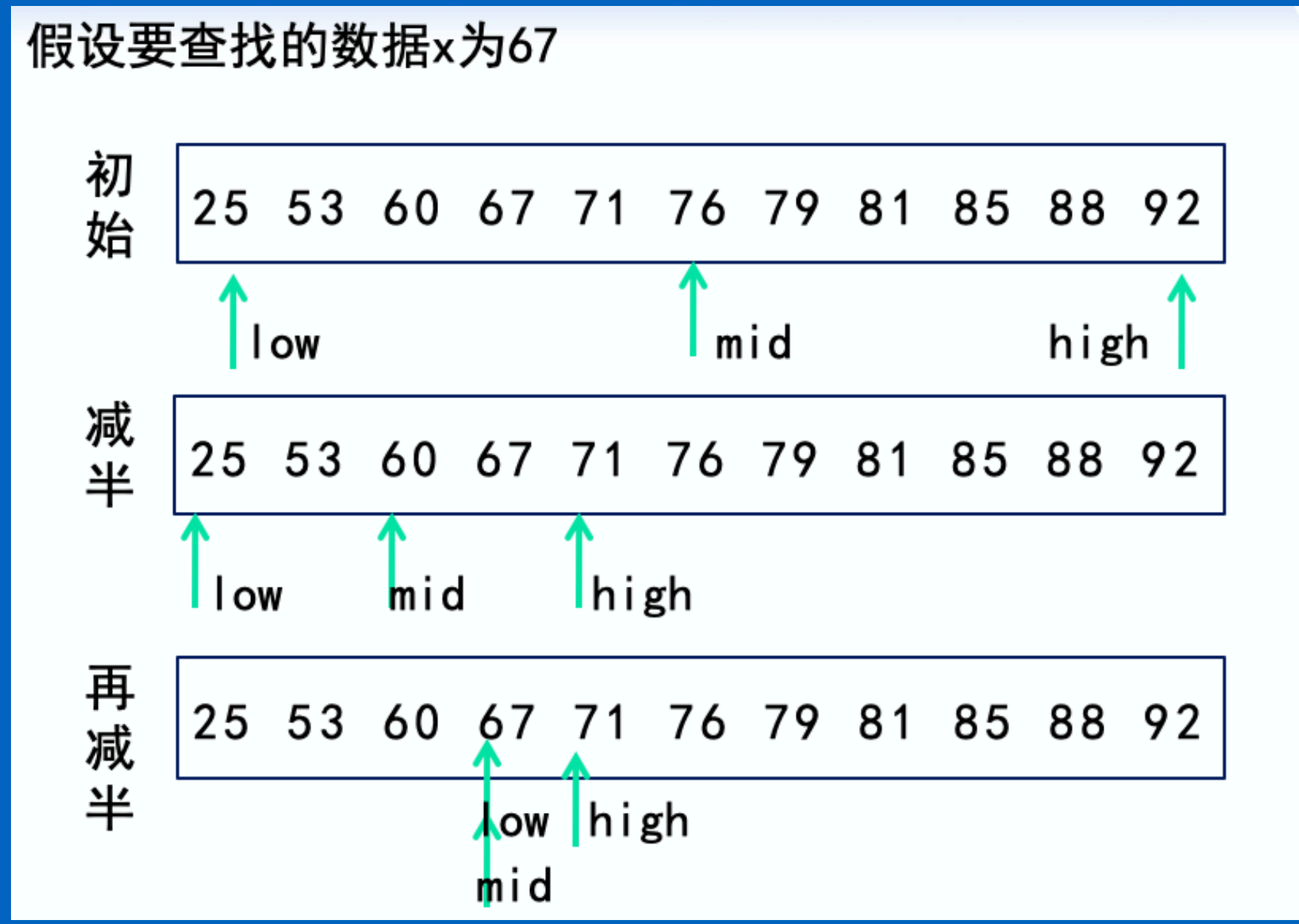
- 基于在有序数组中高效查找的一种常见算法
- 算法思想：
  - 假设low和high是查找区间的起终点下标，则初始状态下 low = 0, high = n - 1;
  - 求待查区间中间元素的下标 mid = (low + high) / 2
  - 通过 a[mid] 和 x 比较的结果决定后续查找范围；
  - 若x == a[mid]，则查找完毕，结束查找过程；
  - 若x < a[mid]，则只需再查找 a[mid] 前面的元素，修改区间上界 high = mid -1；
  - 若x > a[mid]，则只需再查找 a[mid] 后面的元素，修改区间下界 low = mid + 1；
  - 重复以上过程，直到找到 x ； 或再无查找区域（low>high）



© 东华大学

二分法查找

```c
#include <stdio.h>
#define SIZE 100
int main (){
    int  low,high,mid,x,n,a[SIZE];
    // 初始化 a[], n, x,...

    low = 0;
    high = n-1;
    while(low <= high){
        mid = (low + high) / 2;
        if (x == a[mid])
            break;
        else if (x > a[mid])
            low = mid + 1;
        else
            high = mid-1;
    }

    if(low > high)
        printf("未找到\n");
    else
        printf("%d ",mid);

    return 0;
}
```

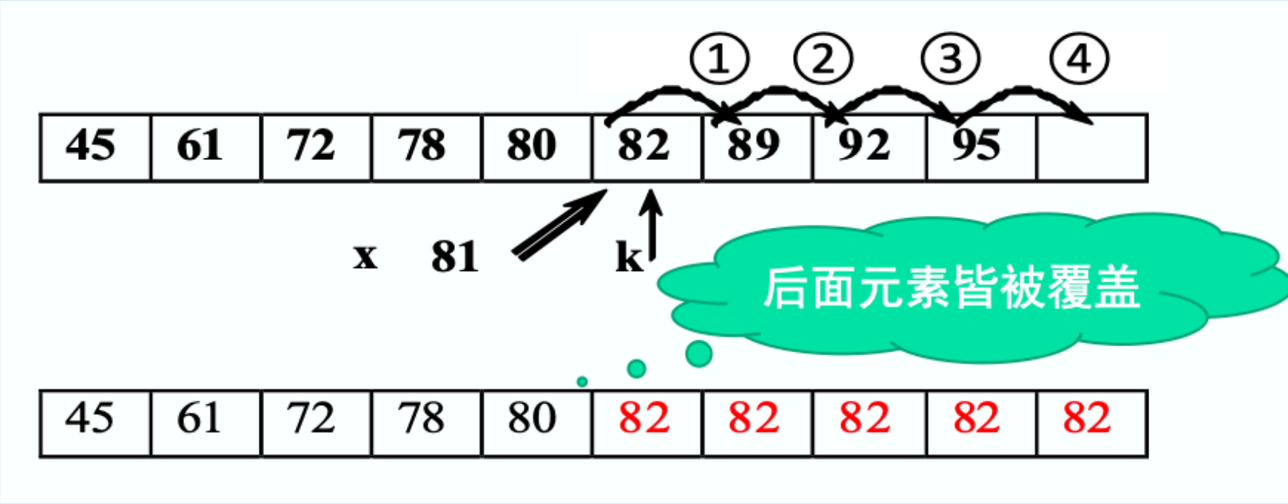


© 东华大学

插入元素

- 在有序数组中，插入一个新元素，使之仍然有序
- 算法思想：
  - 查找待插入数据在数组中应插入的位置k；
  - 从最后一个元素开始向前直到下标为k的元素依次往后移动一个位置；
  - 将欲插入的数据x插入第k个元素的位置
- 从第k个元素开始向后移动?

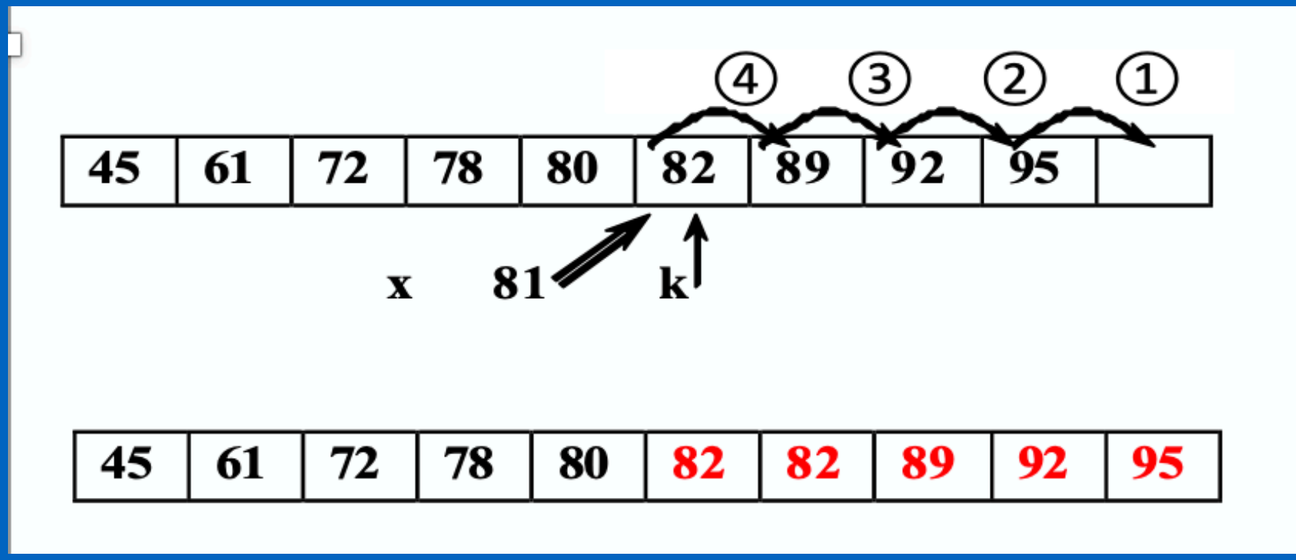
```
for(i = k;i < n;i++)
    a[i+1] = a[i];
```



© 东华大学

插入元素

```c
#include <stdio.h>
#define SIZE 100
int main (){
    int  i,k,n,x,a[SIZE];
    // 初始化 a[],n, x, ...

    for(i = 0; i < n; i++)
        if(x < a[i])
            break ;

    k = i;
    for(i = n-1; i >= k; i--)
        a[i+1] = a[i];
    a[k] = x;

    for(i = 0; i< n+1; i++)
        printf("%3d",a[i]);

    return 0;
}
```

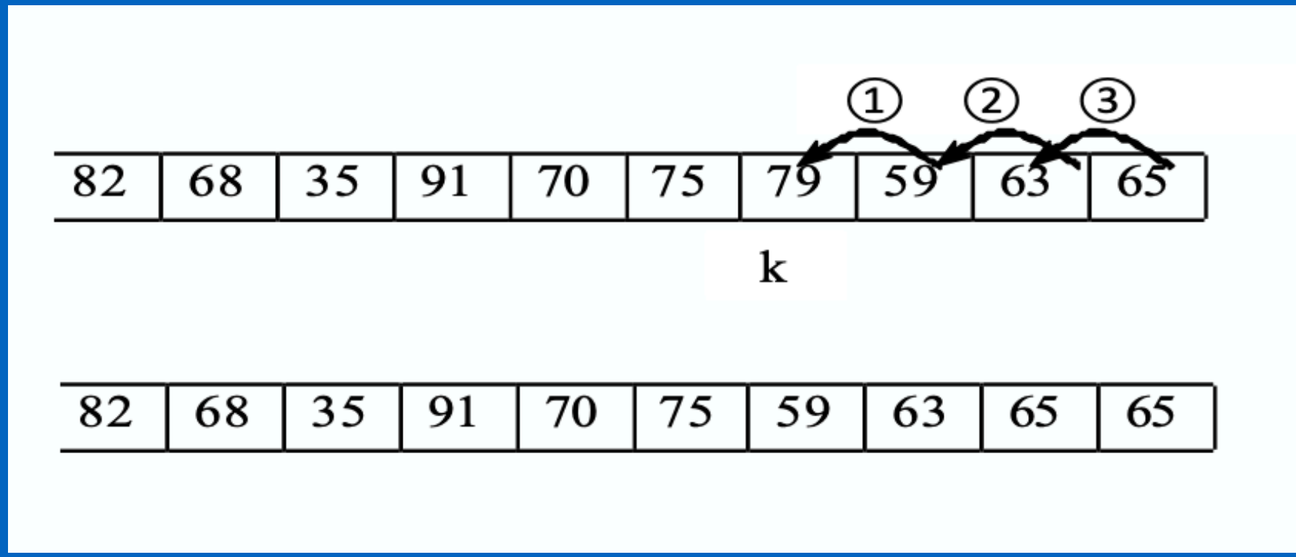


© 东华大学

删除元素

- 例：查找某成绩x是否存在于成绩数组中，若存在，删除第一次出现的该成绩；否则提示“未找到”。

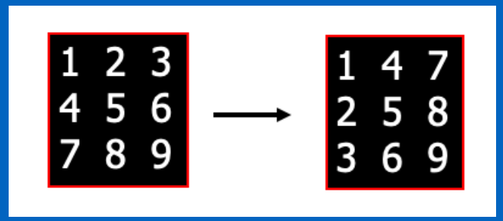
```c
#include <stdio.h>
#define SIZE 100
int main (){
    int  i,k,n,x,a[SIZE];
    // 初始化 a[],n, x, ...

    for(i = 0; i < n;i++)
        if(x == a[i])
            break ;

    k = i;
    for(i = k;i < n-1; i++)
        a[i] = a[i+1];

    a[k] = x;

    for(i = 0; i < n-1; i++)
        printf("%d",a[i]);

    return 0;
}
```



© 东华大学

矩阵转置

- a[i][j] <-> a[j][i]

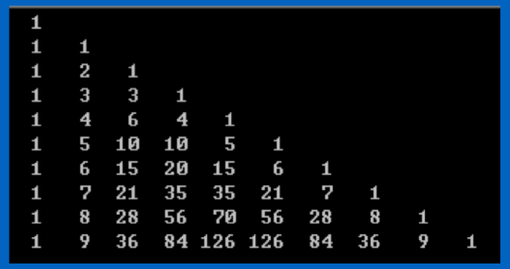



```c
#include <stdio.h>
#define M 3
int main (){
    int a[M][M], b[M][ ], i, j;

    for ( i = 0; i < M; i++)
        for( j = 0; j < M; j++){
            printf("input data ");
            scanf("%d",&a[i][j]);
        }

    for (i = 0; i < M; i++)
        for(j = 0; j < M; j++)
            b[i][j] = a[j][i];

    for (i = 0; i < M; i++){
        for(j = 0; j < M; j++)
            printf("%3d",b[i][j]);
        printf("\n");
    }

    return 0;
}
```

© 东华大学

打印杨辉三角形

- 打印 n 行杨辉三角形
  - 首列 a[i][0] 及主对角线a[i][i] 元素均为1
  - 除此之外的第i行第j列元素的值应为其上一行的当前列与前一列两个元素之和




```c
#include <stdio.h>
#define N 100
int main (){
    int i,j,n,a[N][N];
    printf("请输入行数:");
    scanf("%d",&n);
    //计算首列和主对角线元素:
    for(i = 0; i < n; i++)
        a[i][0] = a[i][i] = 1;
    //计算其他元素:
    for(i = 2; i < n; i++)
        for(j = 1; j < i; j++)
            a[i][j]=
                a[i-1][j-1]+a[i-1][j];
    //打印
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }

    return 0;
}
```

© 东华大学