程序控制结构

南京大学

PYTHON之算法训练——排序与查找

排序

- 在计算机领域里，排序和查找是两种最基本的算法。
- 一个排序算法是能将资料依照特定排序方式排列的一种方法。
- 排序的输出必须遵守下列两个原则：
  - 输出结果为递增序列或递减序列
  - 输出结果是原输入的一种排列、或是重组，不允许出现其他数值
- 3种基本的排序方法：
  - 冒泡排序(bubble sort)
  - 选择排序(selection sort)
  - 插入排序(insertion sort)

排序

- **例1.1 将一个列表中的元素用冒泡法升序排序。**
- 冒泡排序(bubble sort) 算法思想：
  - S1: 首先将所有待排序的数放入一个一个列表中；
  - S2: 从列表的第一个元素开始，依次将相邻两个元素的值进行比较，若前一个元素大于下一个元素，则将它们互相交换。
  - S3: 重复S2步骤，直至再也没有交换为止。
- 对n个数升序排序最差情况是原序列为降序序列，此时要排n-1趟，其中第j趟中要进行n-j次比较。

冒泡法排序

3	3	3
5	5	5
11	7	6
7	6	7
6	11	11
第一趟	第二趟	第三趟

冒泡法排序

第几趟	i	比较次数	j a[j]>a[j+1]
1	0	4次	0~3
2	1	3次	0~2
3	2	2次	0~1
4	3	1次	0

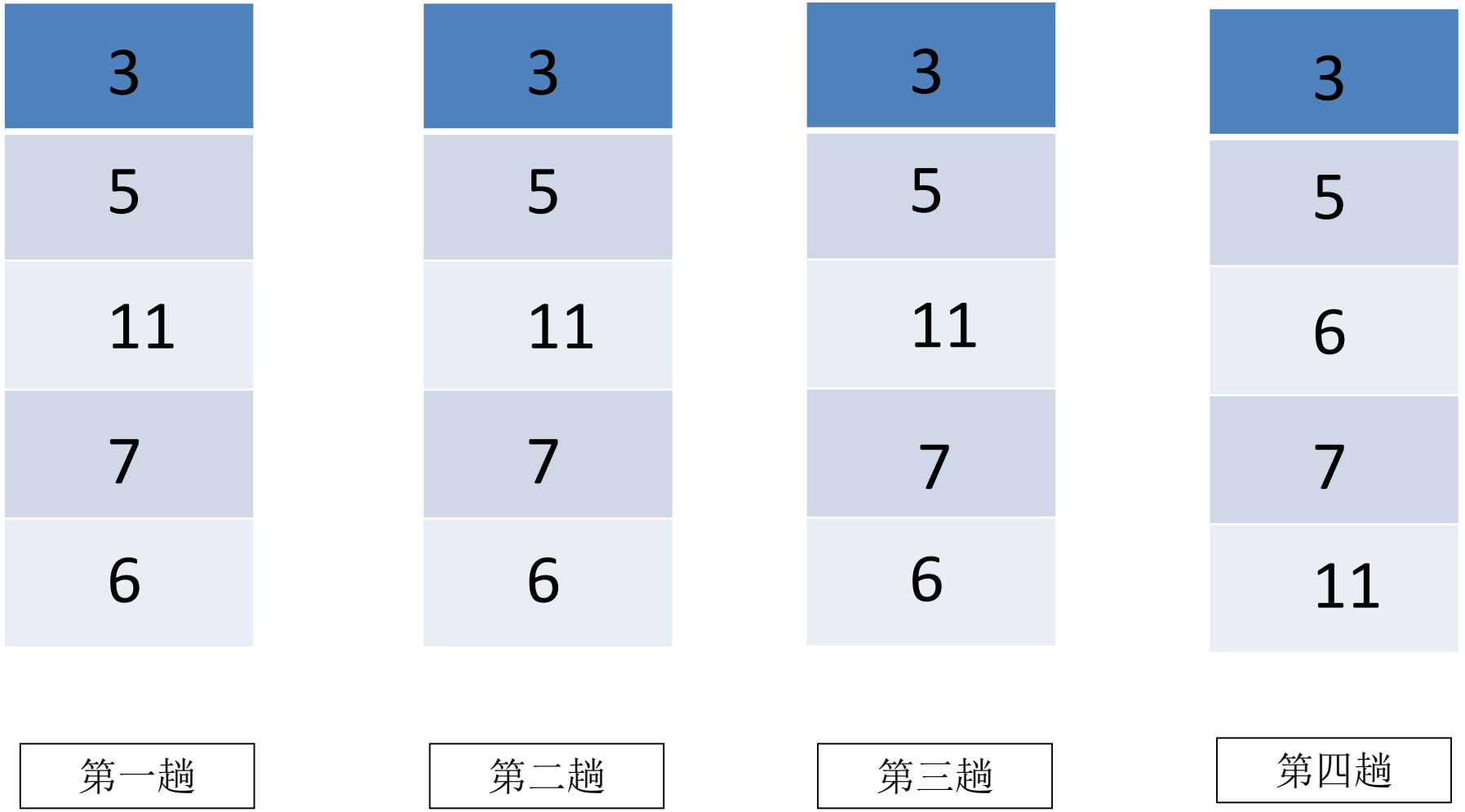
冒泡法排序

```
lst = [3, 5, 11, 7, 6]
n = len(lst)
for i in range(n-1):
    flag = 1
    for j in range(n-1-i):
        if lst[j] > lst[j+1]:
            lst[j], lst[j+1] = lst[j+1], lst[j]
            flag = 0
    if flag:
        break
print(lst)
```

选择排序

- **例1.2 将一个列表中的元素用选择法升序排序。**
- 选择法升序排序算法：
  - S1：设列表中放了n个待排序的数；
  - S2：i=0；
  - S3：从列表的第i+1号元素开始到第n-1号元素，寻找最小元素的下标，并记录；
  - S4：将上一步找到的最小元素和第i号元素交换；
  - S5：使i增1，如果i等于n-1，算法结束，否则回到第S3步。

选择法排序



选择法排序

```
lst = [3, 5, 11, 7, 6]
n = len(lst)
for i in range(n-1):
    point = i
    for j in range(i+1, n):
        if lst[point] > lst[j]:
            point = j
    if point != i:
        lst[i], lst[point] = lst[point], lst[i]
print(lst)
```

以下排序算法是否正确?

```
lst = [3, 5, 11, 7, 6]
n = len(lst)
for i in range(n-1):
    min_lst = lst[i]
    for j in range(i+1, n):
        if lst[j] < min_lst:
            min_lst = lst[j]
    lst[i], min_lst = min_lst, lst[i]
print(lst)
```

插入排序

- **例1.3 将一个列表中的元素用插入法升序排序。**
- 插入法升序排序算法：
  - S1: 首先新建一个空列表，用于保存已排序的有序数列；
  - S2: 从原列表中取出一个数，将其插入“有序列表”中，使其仍旧保持有序状态；
  - S3: 重复S2步骤，直至原列表为空。

插入法排序

3	3	3	3
5	5	5	5
11	11	11	7
7	7	7	11
6	6	6	6
第一趟	第二趟	第三趟	第四趟

插入法排序

- 已经从小到大排好i个元素，现将第i+1个元素（即第i号元素）插入合适的位置，使其仍旧保持有序状态。
  - 第一步，查找合适的位置（下标j），满足条件

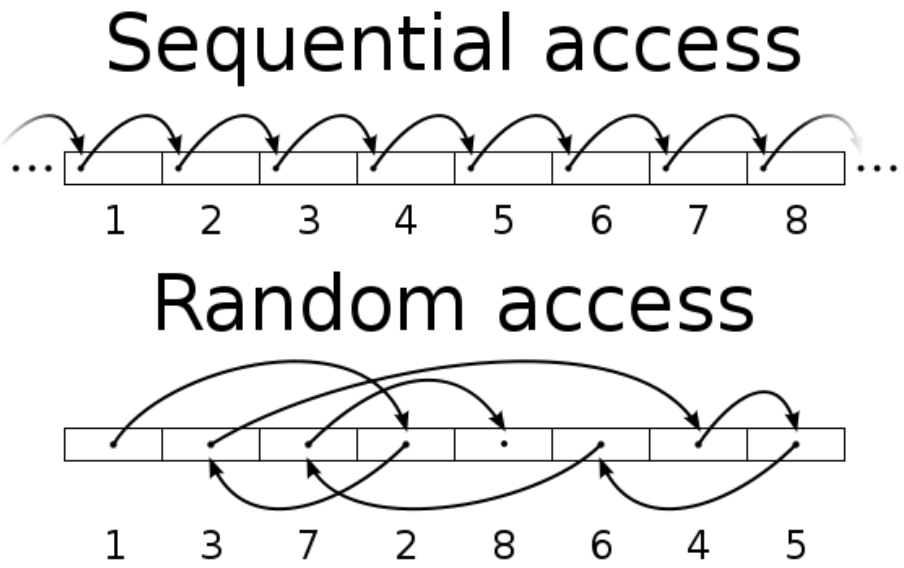
    a[j-1]<=a[i]&& a[i]<=a[j];
  - 第二步，插入元素a[i]，将a[j]至a[i-1]的所有元素后退一个元素位置，把a[j]空出以便存放待插入的值。
- 后退一个元素位置的操作也可以和查找合适位置合并一起操作，即查到合适位置之前都把元素向后退一个元素位置。

插入法排序

```
lst = [3, 5, 11, 7, 6]
n = len(lst)
for i in range(1, n):
    temp = lst[i]
    j = i - 1
    while j >= 0 and temp < lst[j]:
        lst[j+1] = lst[j]
        j -= 1
    lst[j+1] = temp
print(lst)
```

查找

- **查找算法**



- 线性查找法（sequential search）
- 折半查找法（binary search）

线性查找

- **例2.1** 使用**线性查找法**在列表中查找指定数据在列表中的位置。

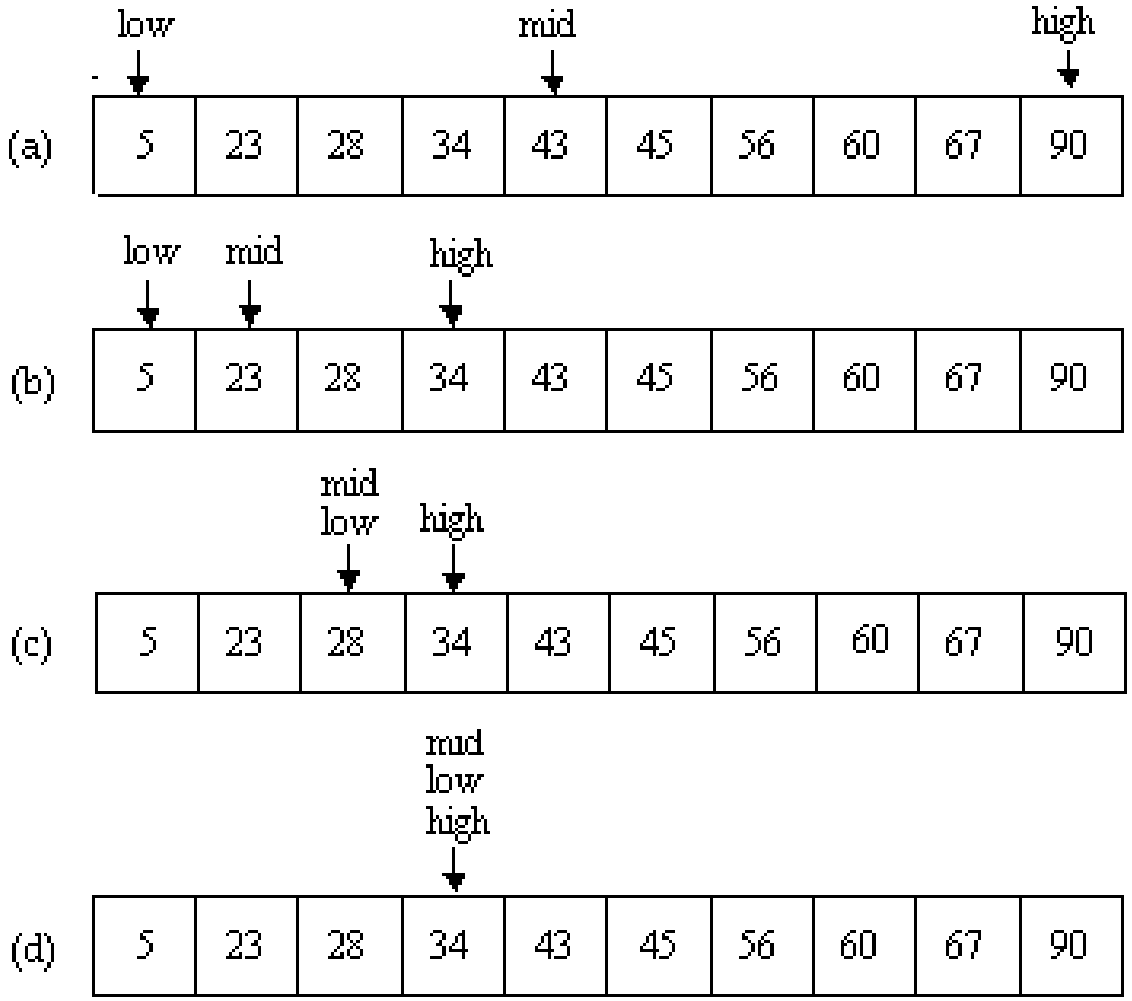
```
lst = [45, 67, 23, 43, 45, 34, 5, 23, 64, 56]
n = len(lst)
m = -1
key = int(input('enter the key: '))
for i in range(n):
    if lst[i] == key:
        m = i
        break
if m != -1:
    print('found!', m)
else:
    print('not found!')
```

查找

- **例2.2** 使用**折半查找法**在有序列表中查找指定数据在列表中的位置。

折半查找法

- 在列表（5，23，28，34，43，45，56，60，67，90）中寻找34。



折半查找法

```
lst = [5, 23, 28, 34, 43, 45, 56, 60, 67, 90]
key = int(input('input the key: '))
low, high = 0, len(lst)-1
while low <= high:
    mid = (low + high) // 2
    if lst[mid] == key:
        k = mid
        print('found: ', mid)
        break
    if key < lst[mid]:
        high = mid -1
    else:
        low = mid + 1
else:
    print('not found!')
```