数据结构与算法（二）

张铭 主讲

采用教材：张铭，王腾蛟，赵海燕 编写
高等教育出版社，2008.6（“十一五”国家级规划教材）

https://pkumooc.coursera.org/bdsalgo-001/

第二章 线性表

- 2.1 线性表
- 2.2 顺序表 

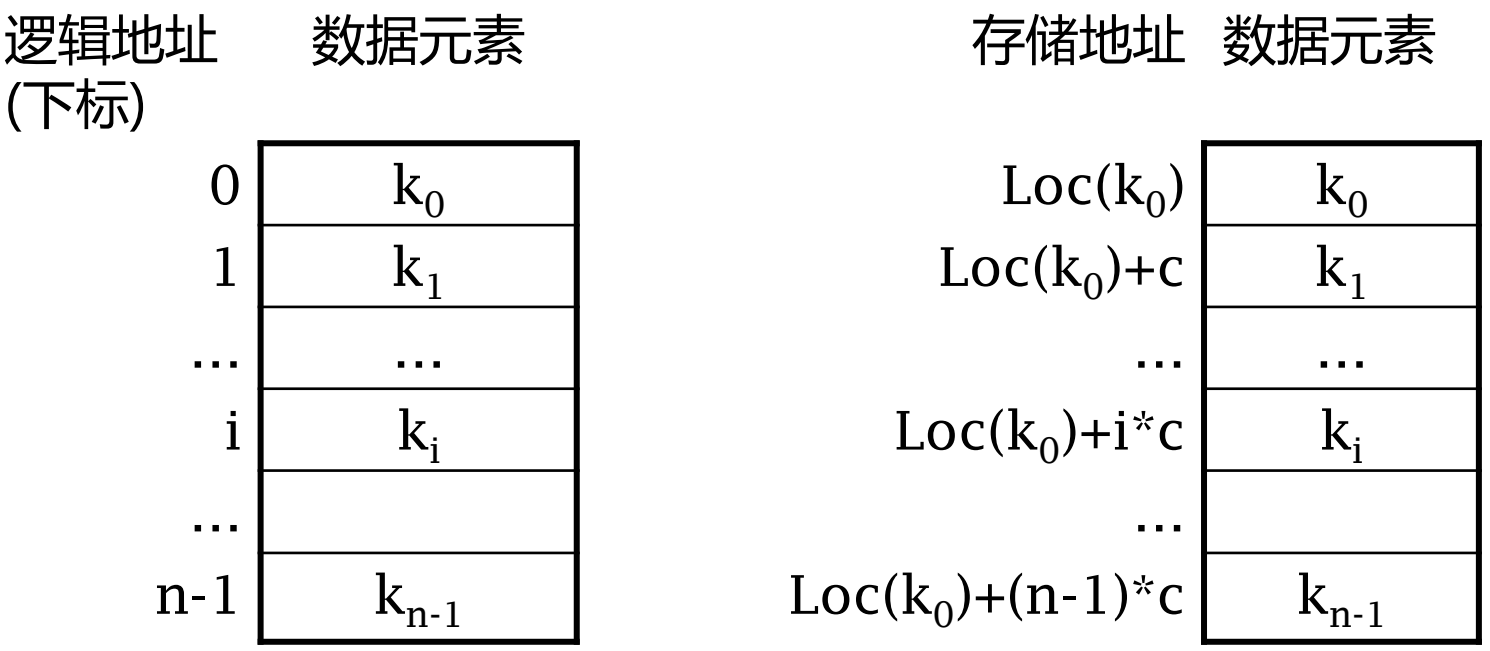
- 2.3 链表
- 2.4 顺序表和链表的比较

2.2 顺序表

- 也称向量，采用定长的一维数组存储结构
- 主要特性
  - 元素的类型相同
  - 元素顺序地存储在连续存储空间中，每一个元素有唯一的索引值
  - 使用常数作为向量长度
- 数组存储
- 读写其元素很方便 ，通过下标即可指定位置
  - 只要确定了首地址，线性表中任意数据元素都可以随机存取

2.2 顺序表

- 元素地址计算如下所示：
  - $Loc(k_i) = Loc(k_0) + c \times i,\ c = \text{sizeof}(ELEM)$



顺序表类定义

```
class arrList : public List<T> {      // 顺序表，向量
private:                              // 线性表的取值类型和取值空间
    T * aList ;                       // 私有变量，存储顺序表的实例
    int maxSize;                      // 私有变量，顺序表实例的最大长度
    int curLen;                       // 私有变量，顺序表实例的当前长度
    int position;                     // 私有变量，当前处理位置
public:
    arrList(const int size) {         // 创建新表，设置表实例的最大长度
        maxSize = size;  aList = new T[maxSize];
            curLen = position = 0;
    }
    ~arrList() {                      // 析构函数，用于消除该表实例
            delete [] aList;
    }
```

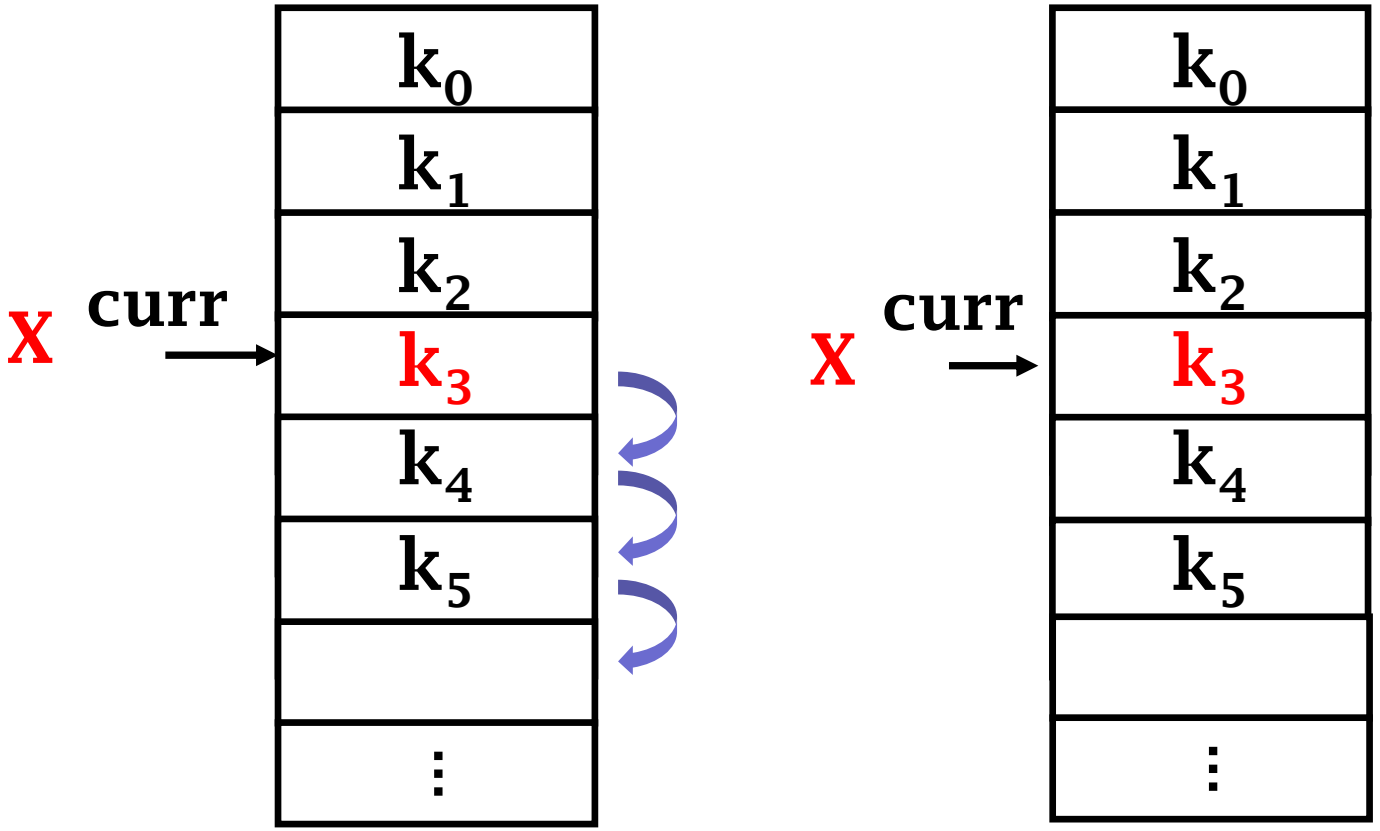
顺序表类定义

```
    void clear() {       // 将顺序表存储的内容清除，成为空表
        delete [] aList; curLen = position = 0;
            aList = new T[maxSize];
    }
    int  length();                                  // 返回当前实际长度
    bool append(const T value);                     // 在表尾添加元素 v
    bool insert(const int p, const T value);        // 插入元素
    bool delete(const int p);                       // 删除位置 p 上元素
    bool setValue(const int p, const T value);      // 设元素值
    bool getValue(const int p, T& value);           // 返回元素
    bool getPos(int &p, const T value);             // 查找元素
};
```

顺序表上的运算

- 重点讨论
  - 插入元素运算
    - bool insert(const int p, const T value);
  - 删除元素运算
    - bool delete(const int p);
- 其他运算请大家思考……

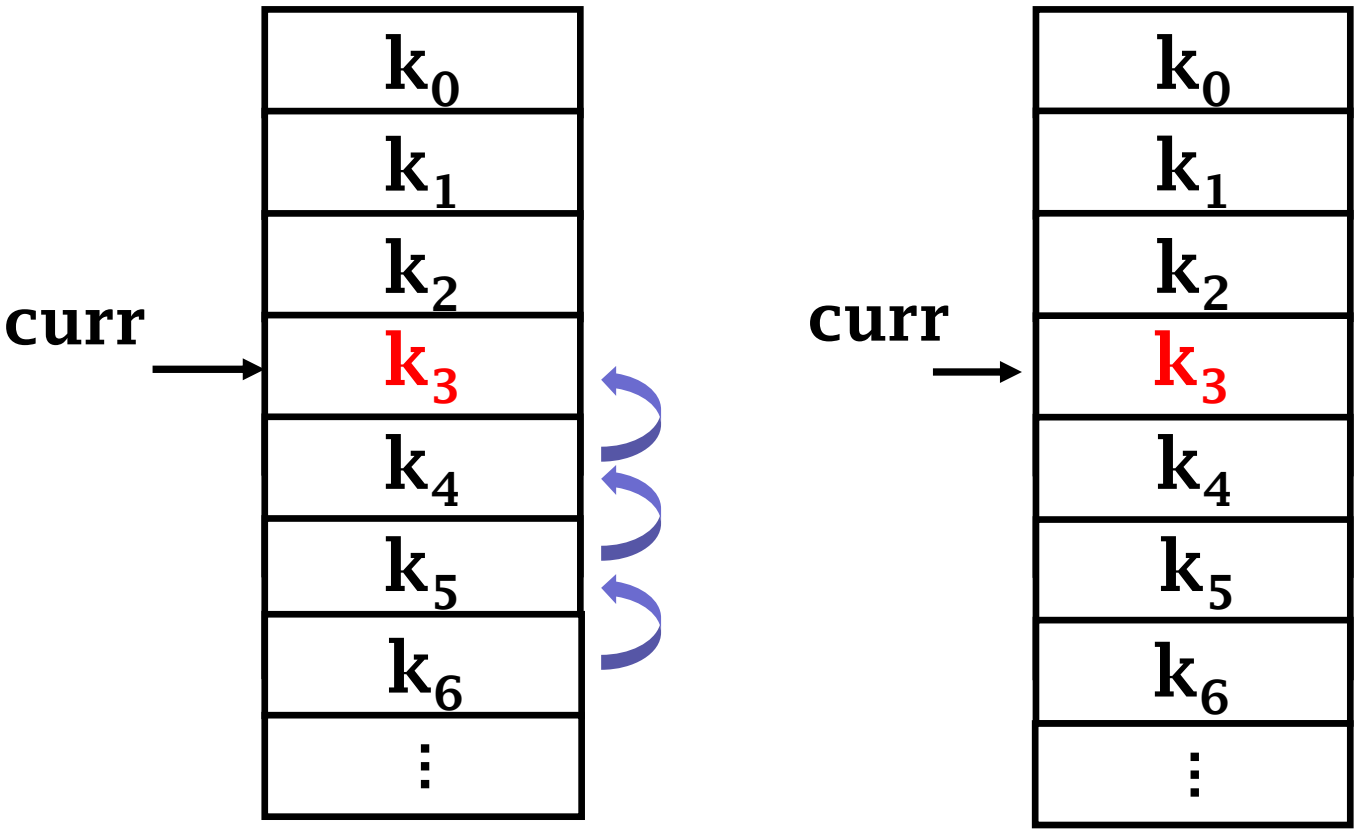
顺序表的插入图示



顺序表的插入

```
// 设元素的类型为T， aList是存储顺序表的数组， maxSize是其最大长度；
// p为新元素value的插入位置，插入成功则返回true， 否则返回false
template <class T>  bool arrList<T> :: insert (const int p, const T value)  {
    int i;
    if (curLen >= maxSize)  { // 检查顺序表是否溢出
        cout << "The list is overflow"<< endl;  return false;
    }
    if (p < 0 || p > curLen)  {  // 检查插入位置是否合法
        cout << "Insertion point is illegal"<< endl;  return false;
    }
    for (i = curLen; i > p; i--)
        aList[i] = aList[i-1];      // 从表尾 curLen -1 起往右移动直到 p
    aList[p] = value;               // 位置 p 处插入新元素
    curLen++;                       // 表的实际长度增 1
    return true;
}
```

顺序表的删除图示



顺序表的删除

```
// 设元素的类型为 T；aList是存储顺序表的数组； p 为即将删除元素的位置
// 删除成功则返回 true，否则返回 false
template <class T>                    // 顺序表的元素类型为 T
bool arrList<T> :: delete(const int p) {
    int i;
    if (curLen <= 0 ) {               // 检查顺序表是否为空
        cout << " No element to delete \n"<< endl;
        return false ;
    }
    if (p < 0 || p > curLen-1) {      // 检查删除位置是否合法
        cout << "deletion is illegal\n"<< endl;
        return false ;
    }
    for (i = p; i < curLen-1; i++)
        aList[i] = aList[i+1];        // 从位置p开始每个元素左移直到 curLen
    curLen--;                         // 表的实际长度减1
    return true;
}
```

顺序表插入删除运算的算法分析

- 表中元素的移动
  - 插入：移动 $n-i$
  - 删除：移动 $n-i-1$ 个
- i 的位置上插入和删除的概率分别是 p_i 和 p_i'
  - 插入的平均移动次数为

$$M_i = \sum_{i=0}^{n}(n-i)p_i$$

  - 删除的平均移动次数为

$$M_d = \sum_{i=0}^{n-1}(n-i-1)p_i'$$

算法分析

- 如果在顺序表中每个位置上插入和删除元素的概率相同，即$p_i = \frac{1}{n+1}, p_i' = \frac{1}{n}$

$$M_i = \frac{1}{n+1}\sum_{i=0}^{n}(n-i) = \frac{1}{n+1}\left(\sum_{i=0}^{n} n - \sum_{i=0}^{n} i\right)$$
$$= \frac{n(n+1)}{n+1} - \frac{n(n+1)}{2(n+1)} = \frac{n}{2}$$
$$M_d = \frac{1}{n}\sum_{i=0}^{n-1}(n-i-1) = \frac{1}{n}\left(\sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i - n\right)$$
$$= \frac{n^2}{n} - \frac{n-1}{2} - 1 = \frac{n-1}{2}$$

时间代价为$O(n)$

思考

- 顺序表中，插入删除操作需要考虑哪些问题？
- 顺序表有哪些优缺点？

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材