Data Structures and Algorithms (2)

Instructor: Ming Zhang

Textbook Authors: Ming Zhang, Tengjiao Wang and Haiyan Zhao

Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

https://courses.edx.org/courses/PekingX/04830050x/2T2014/

Chapter II Linear List

- 2.1 Linear List
- 2.2 Sequential List
- 2.3 Linked List
- 2.4 Comparison between sequential list and linked list

2.2 Sequential List

- **Also known as the vector, fixed-length one-dimensional array is used as the storage structure**
- **Key Features**

  - Elements are of the same type

  - Elements are sequentially stored in contiguous storage space, and each element has a unique index value

  - The type of vector length is constant
- **Implemented as Array**
- **Its elements are easy to read and write, you can specify the location by using its subscript**

  - Once the starting position is got, all the data elements of the list can be random accessed

2.2 Sequential List

. The formula to calculate the elements of location is shown as below:

– $Loc(k_i) = Loc(k_0) + c \times i,\ c = \text{sizeof}(ELEM)$

Logical Address (Subscript)	Data elements
0	k_0
1	k_1
…	…
i	k_i
…	
n-1	k_{n-1}

Store Address	Data elements
$Loc(k_0)$	k_0
$Loc(k_0)+c$	k_1
…	…
$Loc(k_0)+i*c$	k_i
…	
$Loc(k_0)+(n-1)*c$	k_{n-1}

Sequence List's Class Definition

```
class arrList : public List<T> {      // sequential list , vector
private:                              // value types and value space of linear list
   T * aList ;                  // private variables , instance of storage for sequential list
   int maxSize;                 // private variables  , maximum length of the sequential list
   int curLen;                  // private variables  , current length of the sequential list
   int position;                // private variables  , current operation location
public:
   arrList(const int size) {  // construct a new list , set its length to the maximum
      maxSize = size;  aList = new T[maxSize];
         curLen = position = 0;
   }
   ~arrList() {          // destructor function used to eliminate the instance
         delete [] aList;
   }
```

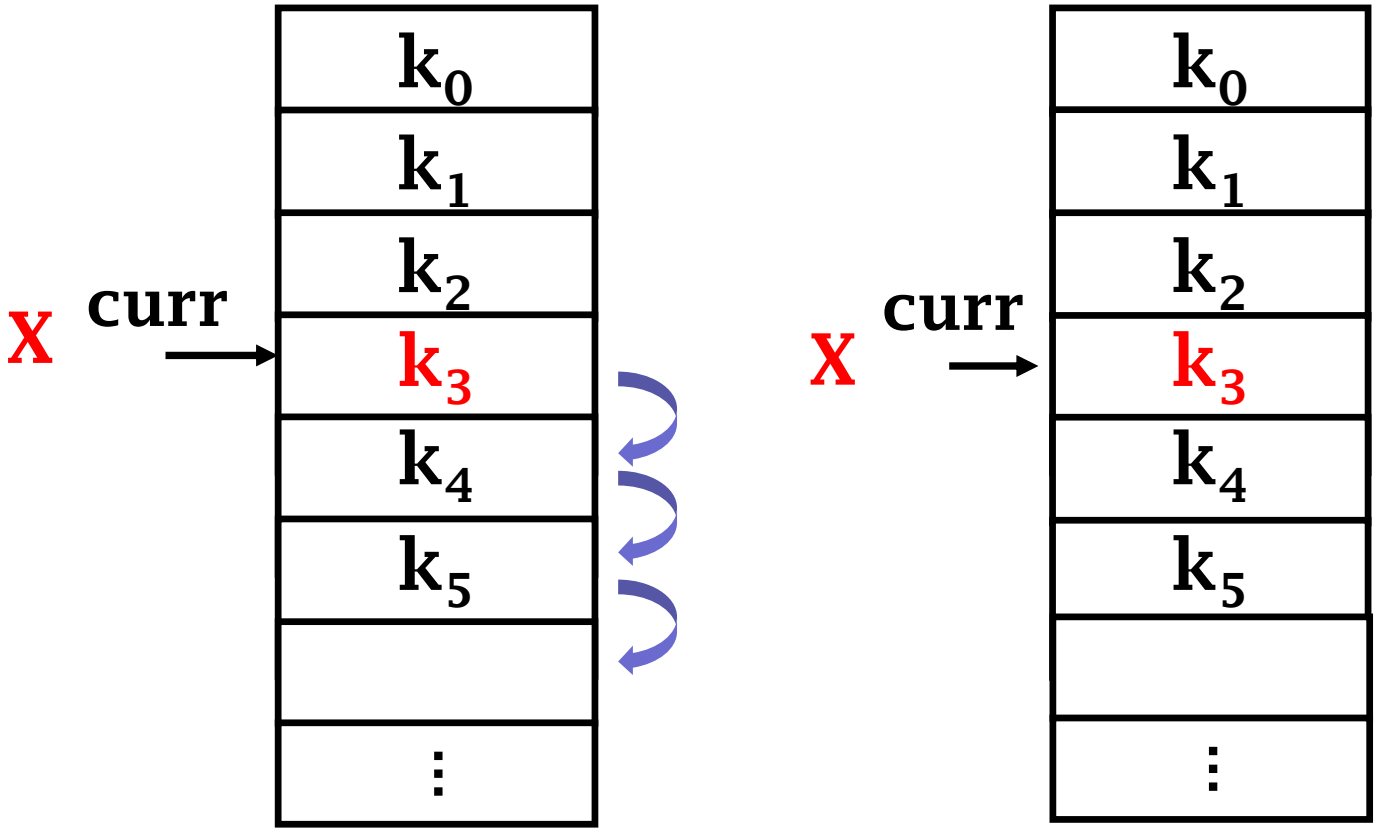
Sequence List's Class Definition

```
    void clear() {       // delete the content , becoming an empty list
          delete [] aList; curLen = position = 0;
          aList = new T[maxSize];
    }
    int  length();                                  // returns the current length
    bool append(const T value);                     // append element v at end
    bool insert(const int p, const T value);        // insert an element at P
    bool delete(const int p);                       // delete the element at P
    bool setValue(const int p, const T value);      // set the value of an element
    bool getValue(const int p, T& value);           // return the value of an element
    bool getPos(int &p, const T value);             // seek for an element
};
```

Operations in Sequential List

- Key discussions
  - Insert element operation
    - bool insert(const int p, const T value);
  - Delete element operation
    - bool delete(const int p);
- Others (Think by yourselves)

Diagram for the insertion of sequential list

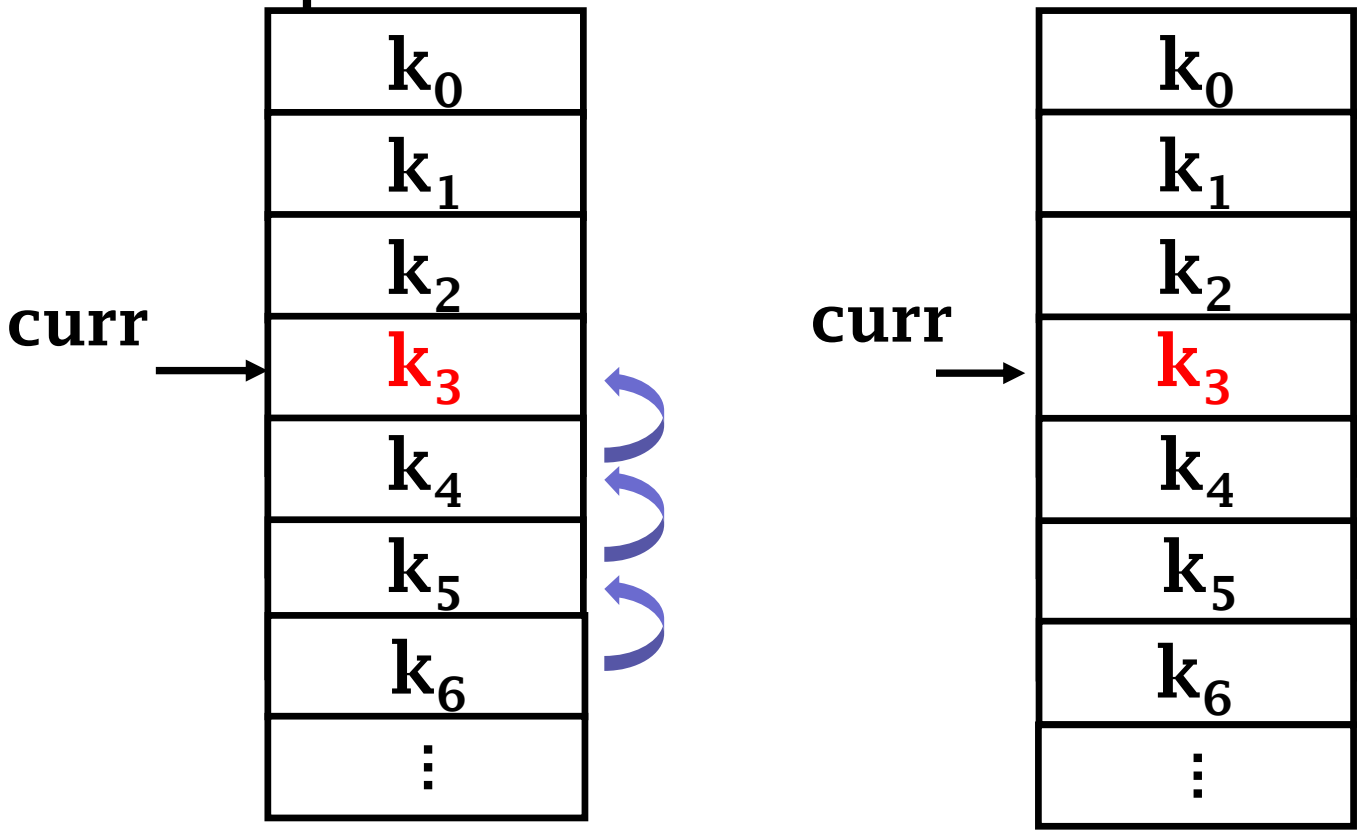


Insertion of sequential list

```
// set the element type as T ,  aList is the array to store Sequential list ,
// maxSize is its maximum length ;
// p is the insert location of the new element  , return true if succeeds ,
// otherwise return false
template <class T>  bool arrList<T> :: insert (const int p, const T value)  {
    int i;
    if (curLen >= maxSize)  { // check if the SL is overflow
        cout << "The list is overflow"<< endl;  return false;
    }
    if (p < 0 || p > curLen)  {  // check if the position to insert is valid
        cout << "Insertion point is illegal"<< endl;  return false;
    }
    for (i = curLen; i > p; i--)
        aList[i] = aList[i-1];    // move right from the end curLen -1 of the
list until p
    aList[p] = value;             // insert a new element at p
    curLen++;                     // adds the current length of the list by 1
    return true;
}
```

Diagram for sequential list's delete operation

- 2.2 Sequential List



Delete operation in sequential list

```
// set the type of the element as T ; aLis is the array to store sequential list
// and p is the position of elements to delete
// returns  true when delete succeed , otherwise returns false
template <class T>                    // the  type of the elements of SL is T
bool arrList<T> :: delete(const int p)  {
    int i;
    if (curLen <= 0 )  {              // Check if the SL is empty
        cout << " No element to delete \n"<< endl;
        return false ;
    }
    if (p < 0 || p > curLen-1)  {     // Check if the position is valid
        cout << "deletion is illegal\n"<< endl;
        return false ;
    }
    for (i = p; i < curLen-1; i++)
        aList[i] = aList[i+1];        // [p, currLen) every element move left
    curLen--;             // the current length of the list decreases by  1
    return true;
}
```

Algorithm analysis of insert and delete operations in sequential list

- The movement of elements in the list
  - Insert: move $n-i$
  - Delete: move $n-i-1$
- The probability values to insert or delete in position i are respectively p_i *and* p_i'
  - The average move time for insert operation is

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

  - The average move time of delete operation is

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

Algorithm Analysis

- If the probability to insert or delete in every location in SL is the same, namely $p_i = \frac{1}{n+1}$, $p_i' = \frac{1}{n}$

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

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

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

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

Time cost is $O(n)$

Thinking

- What should you think about when doing insert or delete operations in sequential list ?
- What advantages and disadvantages does sequential list have?

Data Structures and Algorithms

Thanks

the National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)