# Data Structures and Algorithms ( 8 )

Instructor: Ming Zhang
Textbook Authors : Ming Zhang , Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

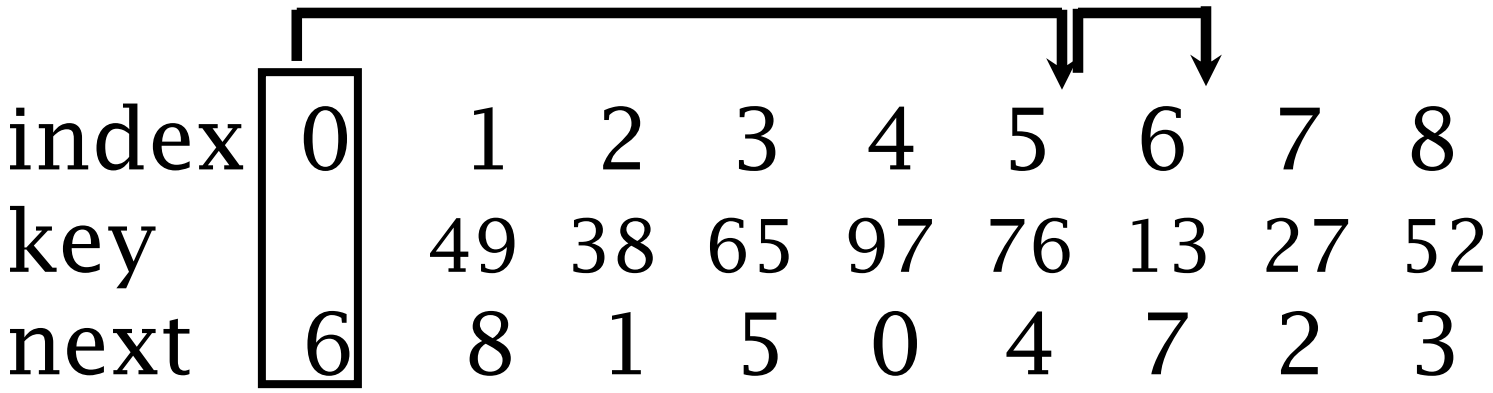
https://courses.edx.org/courses/PekingX/04830050x/2T2014/

# Overview

- 8.1 Basic Concepts of Sorting
- 8.2 Insertion Sort (Shell Sort)
- 8.3 Selection Sort (Heap Sort)
- 8.4 Exchange Sort
  - 8.4.1 Bubble Sort
  - 8.4.2 Quick Sort
- 8.5 Merge Sort
- **8.6** Distributive Sort and **Index Sort**
- 8.7 Time Cost of Sorting Algorithms
- Knowledge Summary on Sorting

## 8.6.3 Index Address Sort

# Index Representation of Result Address of Radix Sort




Insertion algorithm of single linked list with head

Result of linked radix sort

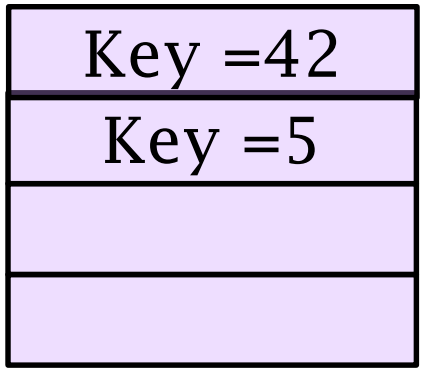
## 8.6.3 Index Address Sort

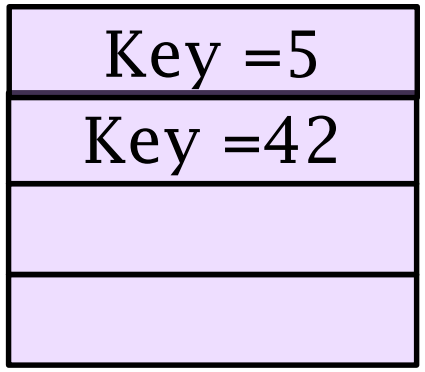
# Handle Static Linked List in Linear Time

```
template <class Record>
void AddrSort(Record *Array, int n, int first)  {
    int i, j;
    j = first;                    // j is the subscript of the  data to be dealt with
    Record TempRec;
    for (i = 0; i < n-1; i++)  {// loop , deal with the ith record every time
        TempRec = Array[j]; // temporarily store the ith record Array[j]
        swap(Array[i], Array[j]);
        Array[i].next = j;      // next chain stores the exchange track j
        j = TempRec.next;  // j moves to the next position
        while (j <= i)          // if j is smaller than i , it is the track , find along the link
            j = Array[j].next;
    }
}
```

## 8.6.3 Index Address Sort

# Index Array

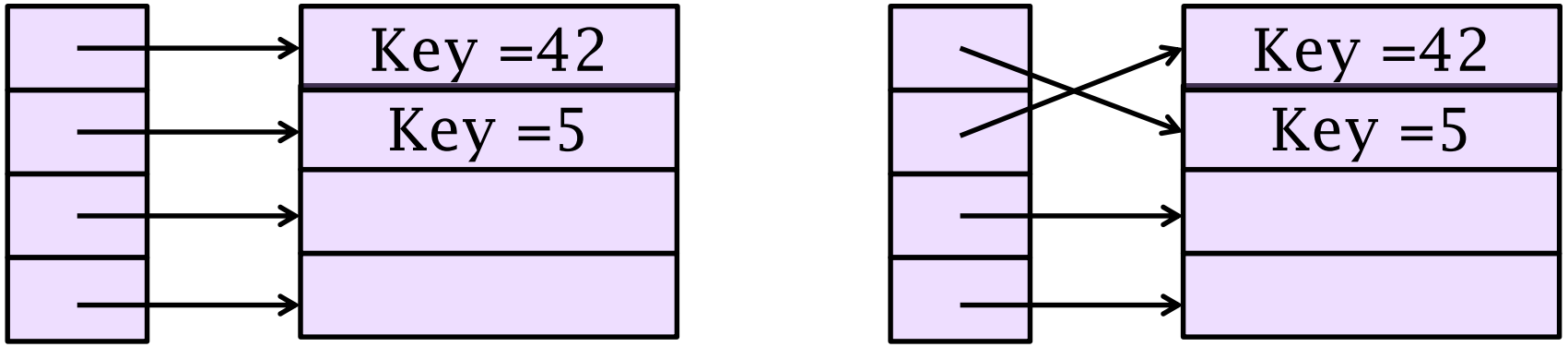



Data field is very large, the cost of exchange records is pretty high

## 8.6.3 Index Address Sort

# Index Array




Exchange pointers to reduce
the times of exchange records

## 8.6.3 Index Address Sort

# Index Result :

- IndexArray[i] stores the position of the data that should be in Array[i]
- After handling, Array[i] is corresponding to Array[IndexArray[i]] in the original array

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Subscript | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Sorting key | 29 | 25 | 34 | 64 | 34’ | 12 | 32 | 45 |
| Result | 5 | 1 | 0 | 6 | 2 | 4 | 7 | 3 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | 12 | 25 | 29 | 32 | 34 | 34’ | 45 | 64 |

# Applicability of Index Sort

- All the ormal sorting methods work
  - Those assignments (or exchanges) are replaced by the assignments (or exchanges) towards the index array
- Example : Insertion Sort

# Index Address Sort Edition of Insertion Sort

```
template<class Record>
void AddrSort(Record Array[], int n) {
    //n is the length of array
    int *IndexArray = new int[n],  TempIndex;
    int i,j,k;
    Record TempRec;          //only need one temporary space
    for (i=0; i<n; i++)
        IndexArray[i] = i;
    for (i=1; i<n; i++)          // insert the ith record in turn
        for (j=i; j>0; j--)      // compare, exchange reversed pairs
            if ( Array[IndexArray[j]] <Array[IndexArray[j-1]]))
                swap(IndexArray, j, j-1);
            else break;          //record before i is in order
```

## 8.6.3 Index Address Sort

# Handle Along Links to Index Array

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Subscript | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Sorting Key | 29 | 25 | 34 | 64 | 34′ | 12 | 32 | 45 |
| Index | 5 | 1 | 0 | 6 | 2 | 4 | 7 | 3 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Result | 12 | 25 | 29 | 32 | 34 | 34′ | 45 | 64 |

# Sort Edition of Insertion Sort Index Address (Cont.)

```
for(i=0;i<n;i++) { // adjust to make subscripts in order
        j= i;
        TempRec = Array[i];
        while (IndexArray[j] != i)  {
            k=IndexArray[j];
            Array[j]=Array[k];
            IndexArray[j] = j;
            j = k;
        }
        Array[j] =TempRec;
        IndexArray[j] = j;
}
}
```

# The Second Index Method

- IndexArray[i] store where data in Array[i] should be
- Sorted Array[IndexArray[i]] is corresponding to Array[i] in the original array

| subscript | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Sorting key | 29 | 25 | 34 | 64 | 34’ | 12 | 32 | 45 |
| Result | 2 | 1 | 4 | 7 | 5 | 0 | 3 | 6 |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | 12 | 25 | 29 | 32 | 34 | 34’ | 45 | 64 |

# Handle Along links for the Second Method

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| · Subscript | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| · Sorting Key | 29 | 25 | 34 | 64 | 34′ | 12 | 32 | 45 |
| · Index | 2 | 1 | 4 | 7 | 5 | 0 | 3 | 6 |

· Result

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 12 | 25 | 29 | 32 | 34 | 34′ | 45 | 64 |

# Thinking

1. Prove the time cost of handling address sort is $\theta(n)$
2. Modify quick sort to get the first index result
3. Get the second index method with Rank sort
4. Get the second index method by simple changing the static linked radix sort

# 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)**