C Programming Basic – week 11

Topics of this week

- Advanced Sorting Algorithm
  1. Quick sort
  2. Merge sort
- Exercises

1. Quicksort

Given an array of *n* elements (e.g., integers):

- If array only contains one element, return
- Else
  - pick one element to use as *pivot.*
  - Partition elements into two sub-arrays:
    - Elements less than or equal to pivot
    - Elements greater than pivot
  - Quicksort two sub-arrays
  - Return results

Example

- Given array of integers

Quick Sort (Hoare)

- Given $(R_0, R_1, \ldots, R_{n-1})$

  K_i: pivot key

  if K_i is placed in S(i),

  then $K_j \le K_{s(i)}$ for $j < S(i)$,

  $K_j \ge K_{s(i)}$ for $j > S(i)$.

- $R_0, \ldots, R_{S(i)-1}, R_{S(i)}, R_{S(i)+1}, \ldots, R_{S(n-1)}$

two partitions

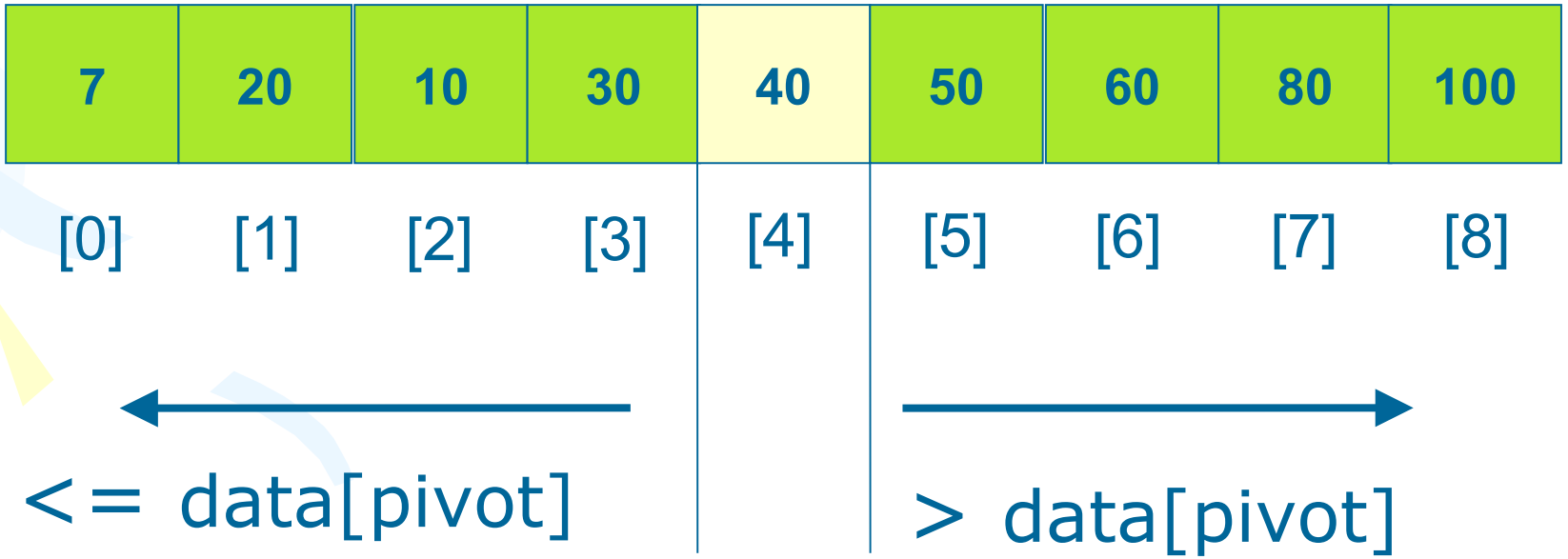
Partitioning

- Given a pivot, partition the elements of the array such that the resulting array consists of:

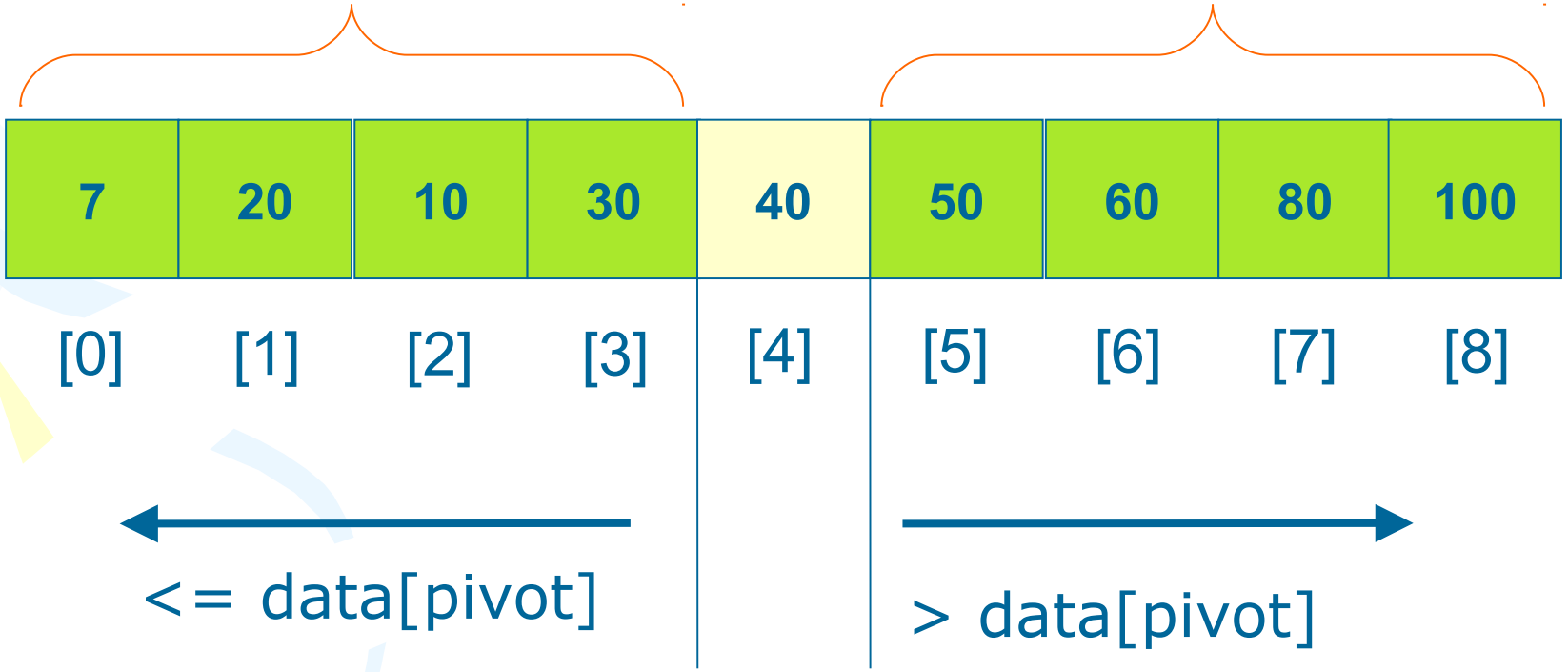
1. One sub-array that contains elements >= pivot
2. Another sub-array that contains elements < pivot

- The two sub-arrays are stored in the original array
- Partitioning by swapping elements

Partitioning example



Recursion



Example

R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	left	right
{26	5	37	1	61	11	59	15	48	19}	0	9
{11	5	19	1	15}	26	{59	61	48	37}	0	4
{1	5}	11	{19	15}	26	{59	61	48	37}	0	1
1	5	11	15	19	26	{59	61	48	37}	3	4
1	5	11	15	19	26	{48	37}	59	{61}	6	9
1	5	11	15	19	26	37	48	59	{61}	6	7
1	5	11	15	19	26	37	48	59	61	9	9
1	5	11	15	19	26	37	48	59	61		

Quick Sort

```
void quicksort(element list[], int left,
                                int right)
{
  int pivot, i, j;
  element temp;
  if (left < right) {
    i = left;     j = right+1;
    pivot = list[left].key;
    do {
      do i++; while (list[i].key < pivot);
      do j--; while (list[j].key > pivot);
      if (i < j) SWAP(list[i], list[j], temp)
    } while (i < j);
    SWAP(list[left], list[j], temp);
    quicksort(list, left, j-1);
    quicksort(list, j+1, right);
  }
}
```

Exercise 11.1

- We assume that you make a mobile phone's address book.
- Declare the structure that can store "name", "phone number" and "e-mail address", and declare an array of size 100 to store records
- Write a program that reads 10 items from an input file to the array and write data to an output file after sorting name in ascending order using Quicksort

Exercise 11.2

- Initiate an array of n random integers. N is entered by user.
- Sort the array using insertion sort and quicksort
- Compare the execution time of two algorithms.
- Run the program with various values of n to view the effect.

Exercise 11.3

- Write a function to select sorting algorithm – If the number of items is more than *x*, quicksort is selected, otherwise, it selects insertion sort
- Note: x is a program argument.
- Read a text file containing more than 100 characters, sort the first 100 characters, and show the result to standard output.

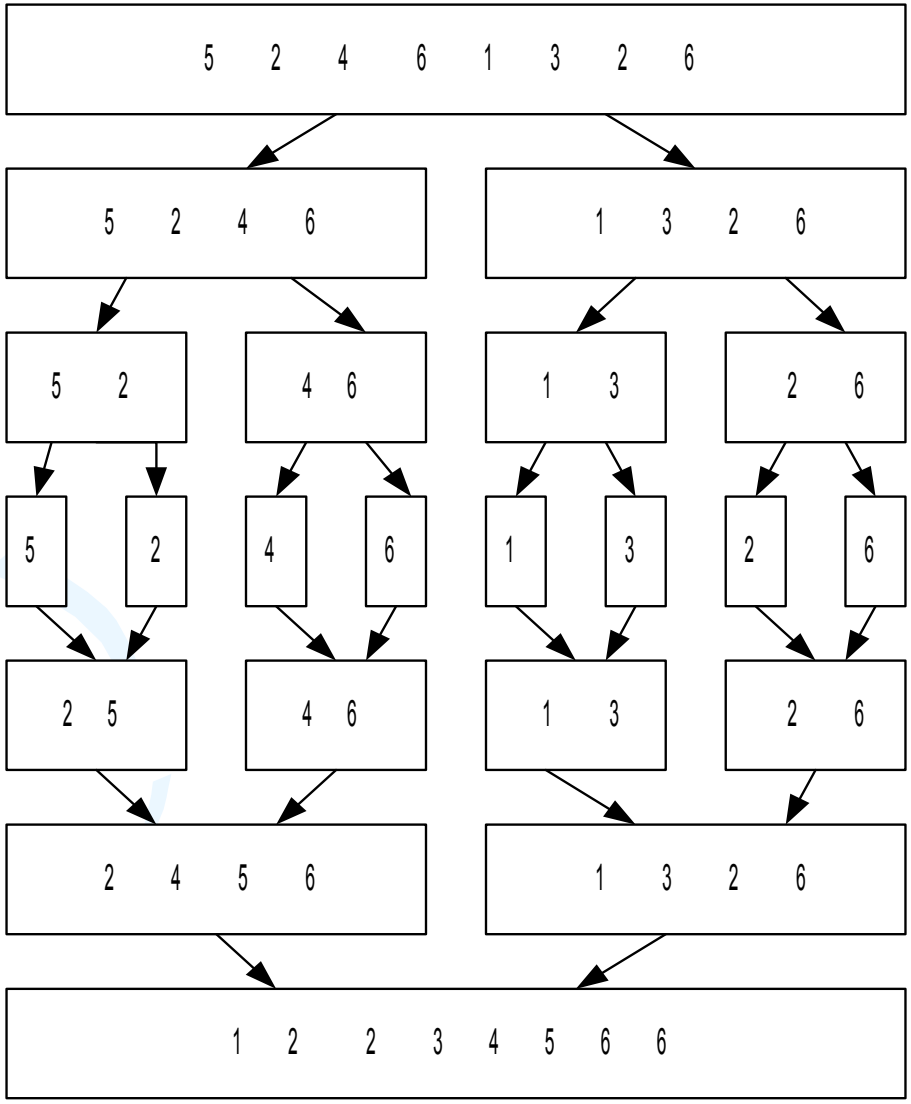
2. Merge Sort

- Problem: Given *n* elements, sort elements into non-decreasing order
- Apply divide-and-conquer to sorting problem
  - If n=1 terminate (every one-element list is already sorted)
  - If n>1, partition elements into two sub-arrays; sort each; combine into a single sorted array

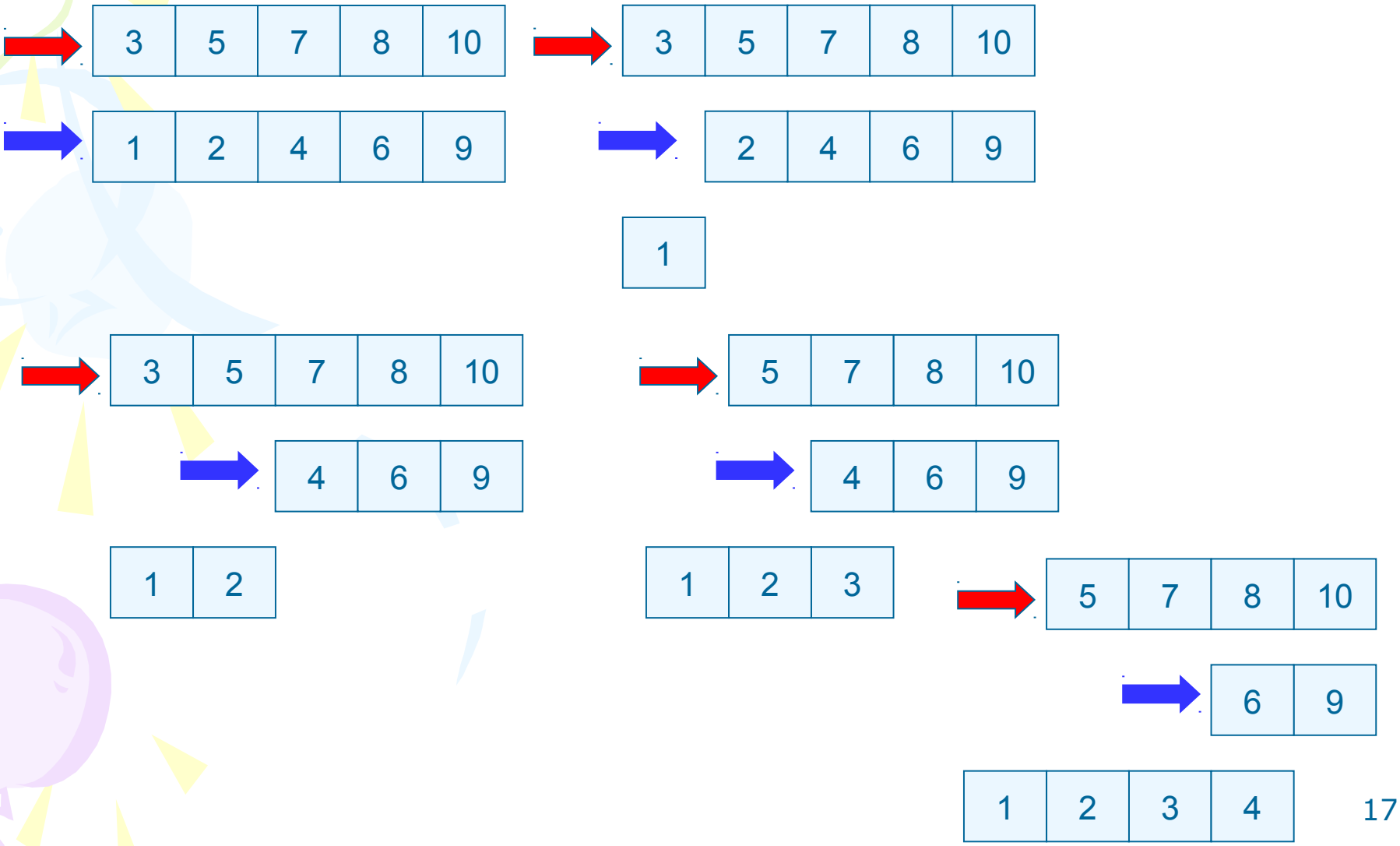
Algorithm

```
MergeSort (E[ 0 .. N])
  if N < threshold
     InsertionSort ( E[0..N] )
  else
     copy E[0.. N/2] to U[0.. N/2]
     copy E[N/2 .. N] to V[0 .. N-N/2]
     MergeSort(U[0 .. N/2])
     MergeSort(V[0 .. N-N/2])
     Merge( U[0 .. N/2], V[0 .. N-N/2},
  E[0 .. N] )
```

Example


5 2 4 6 1 3 2 6
5 2 4 6
1 3 2 6
5 2
4 6
1 3
2 6
5
2
4
6
1
3
2
6
2 5
4 6
1 3
2 6
2 4 5 6
1 3 2 6
1 2 2 3 4 5 6 6

Process of merge



Merge algorithm

```
Merge(U[0..m],V[0..n],E[0..n+m])
  i =  0 ,  j = 0
  k = 0
  while k <  n+m
     if U[i] < V [j]
        E[k] = U[i] , i++
     else
        E[k] = V[j] , j++
     k++
```

Exercise 11.4

- We assume that you make a mobile phone's address book.
- Declare a structure that can store "name", "phone number" and "e-mail address", and declare a singly-linked list to store data
- Write a program that reads 10 items from an input file to the list and writes the data to an output file after sorting in ascending order of name using Merge sort