Royal University of Phnom Penh
Faculty of Engineering

Data Structures and Algorithms

Chapter 3

Recursion and Quicksort

DR. SRUN SOVILA

Outline

- Recursion
- Applied Recursion
- Quicksort
- Improving Quicksort

Outline

- Recursion
- Applied Recursion
- Quicksort
- Improving Quicksort

Quicksort

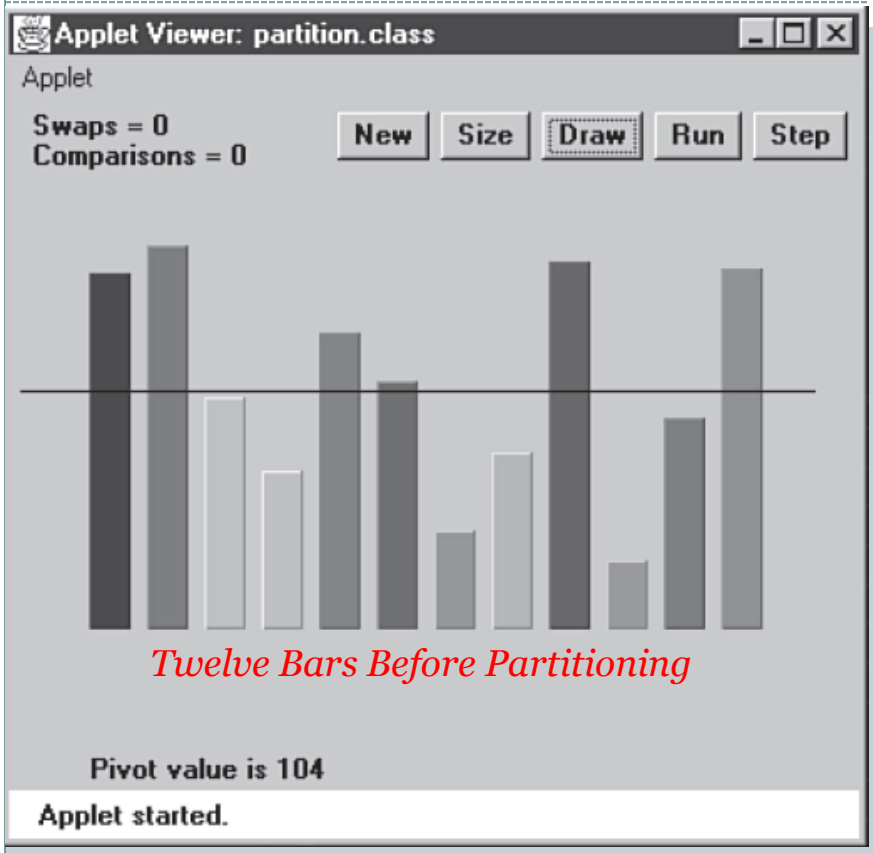
- The bubble and insertion sorts — are easy to implement but are rather slow
- Mergesort is applied recursion, it runs much faster than the simple sorts, but requires twice space as original array

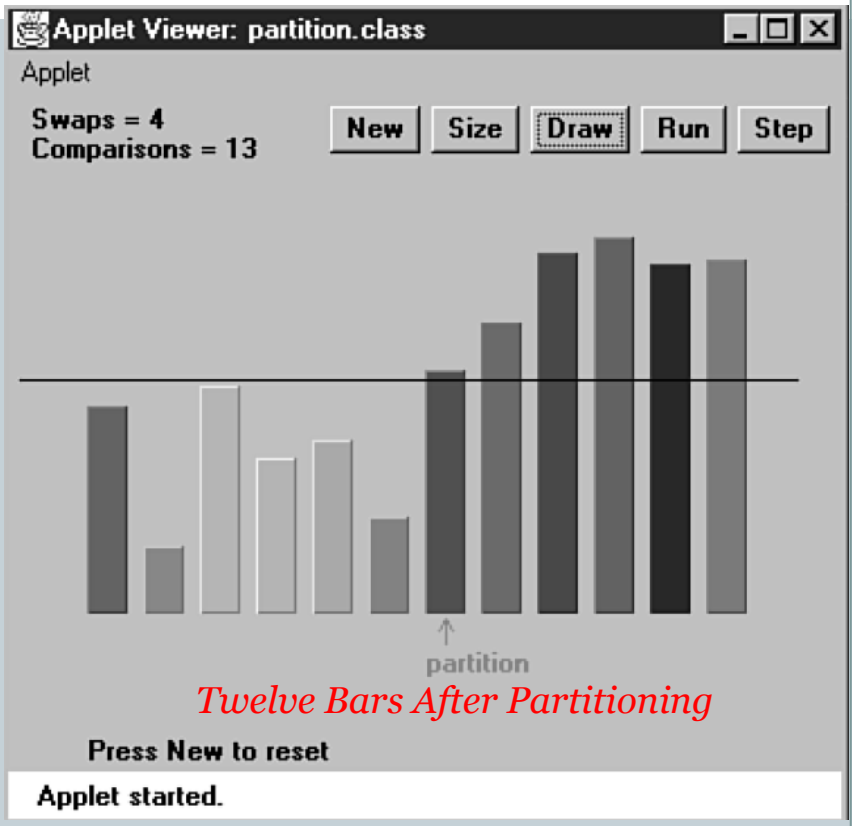
- Quicksort runs faster than simple sorts, in O(N*logN) time, it does not require a large amount of extra memory space, as mergesort
- Quicksort is based on the idea of partitions

Quicksort: Partitioning

- To *partition* data is to divide it into two groups:
  - all the items with a key value higher than a specified amount;
  - All the time with a lower key value.
- Examples
  - Maybe you want to divide your personnel records into two groups: employees who live within 15 miles of the office and those who live farther away
  - A school administrator might want to divide students into those with grade point averages higher and lower than 3.5, so as to know who deserves to be on the dean's list

Quicksort: Partitioning Example





- Pivot – is the value used to determine into two groups (less and greater). It is the border of less than and greater.

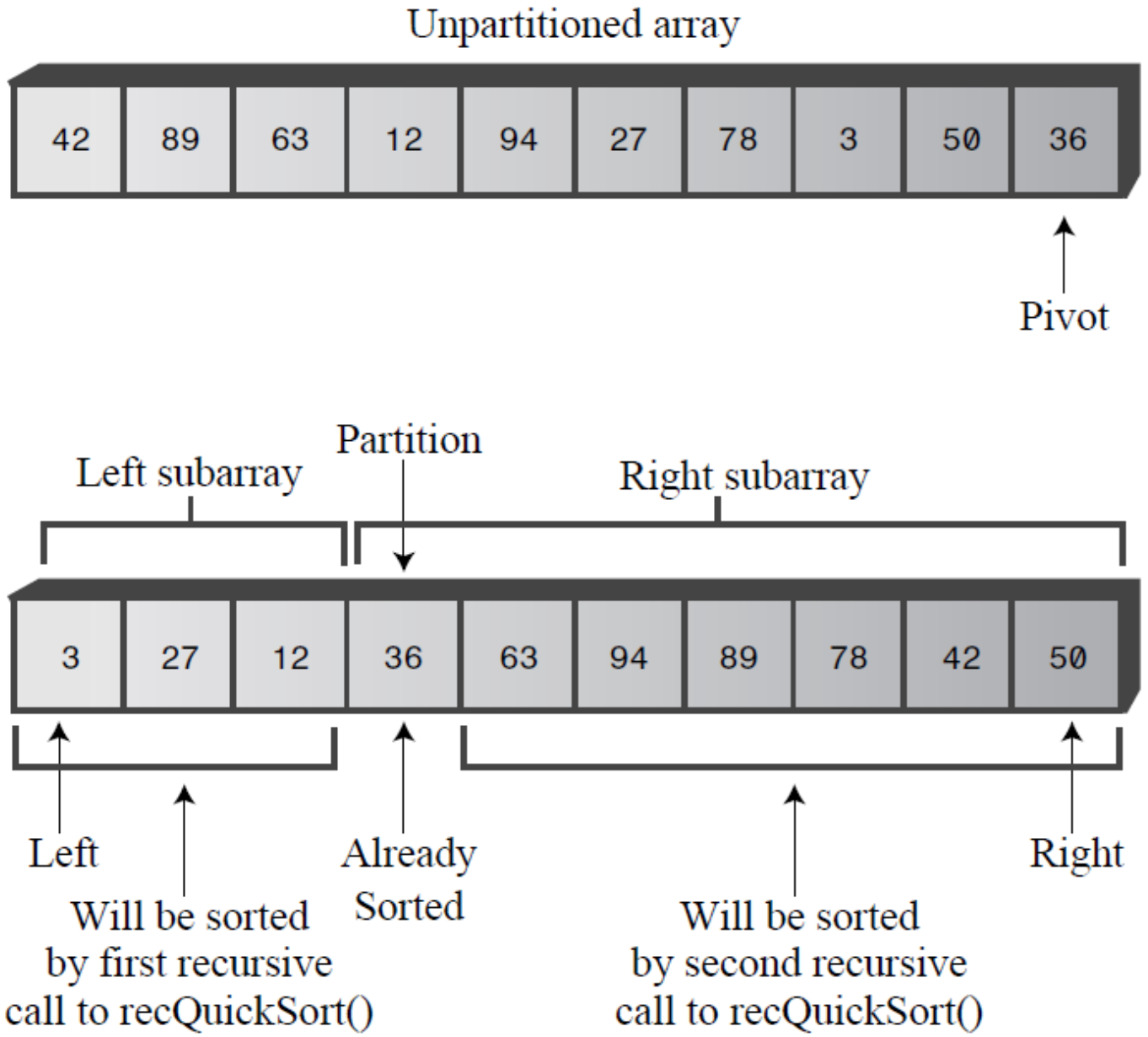
Quicksort: Partitioning Pseudo Code


```
int PartitionIt( left, right, pivot) {
  while( true ){
    find the element greater than pivot; //find to the right, but
                                         //possible greater than right element

    find the element smaller than pivot; //find to the left, but possible
                                         //less than the left element

    if the index of left cross to right //partition done
      then partition done (break);
    else swap( LeftMark, RightMark );
  }
  return LeftMark;
}
```

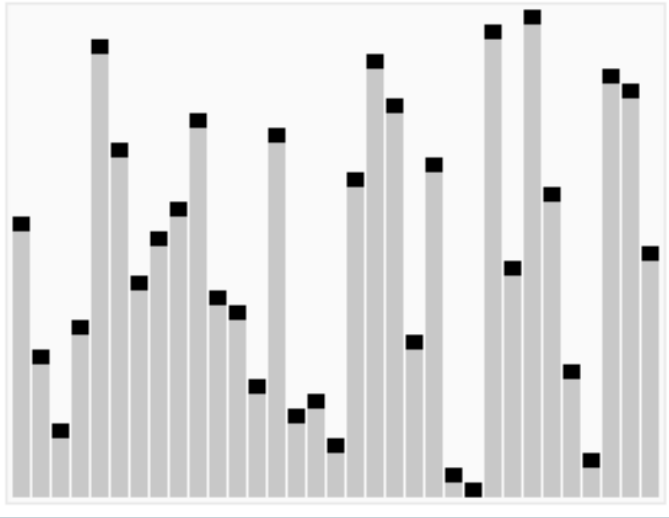
Basic Quicksort

- Quicksort was discovered by British Computer Scientist C. A. R. Hoare, 1962
- Basically the quicksort algorithm operates by partitioning an array into two sub-arrays, and then calling itself recursively to quicksort each of these sub-arrays
- The pivot will be selected at right, but it will be finally placed between these two sub-arrays

Quicksort: Recursive Calls Sort Subarrays

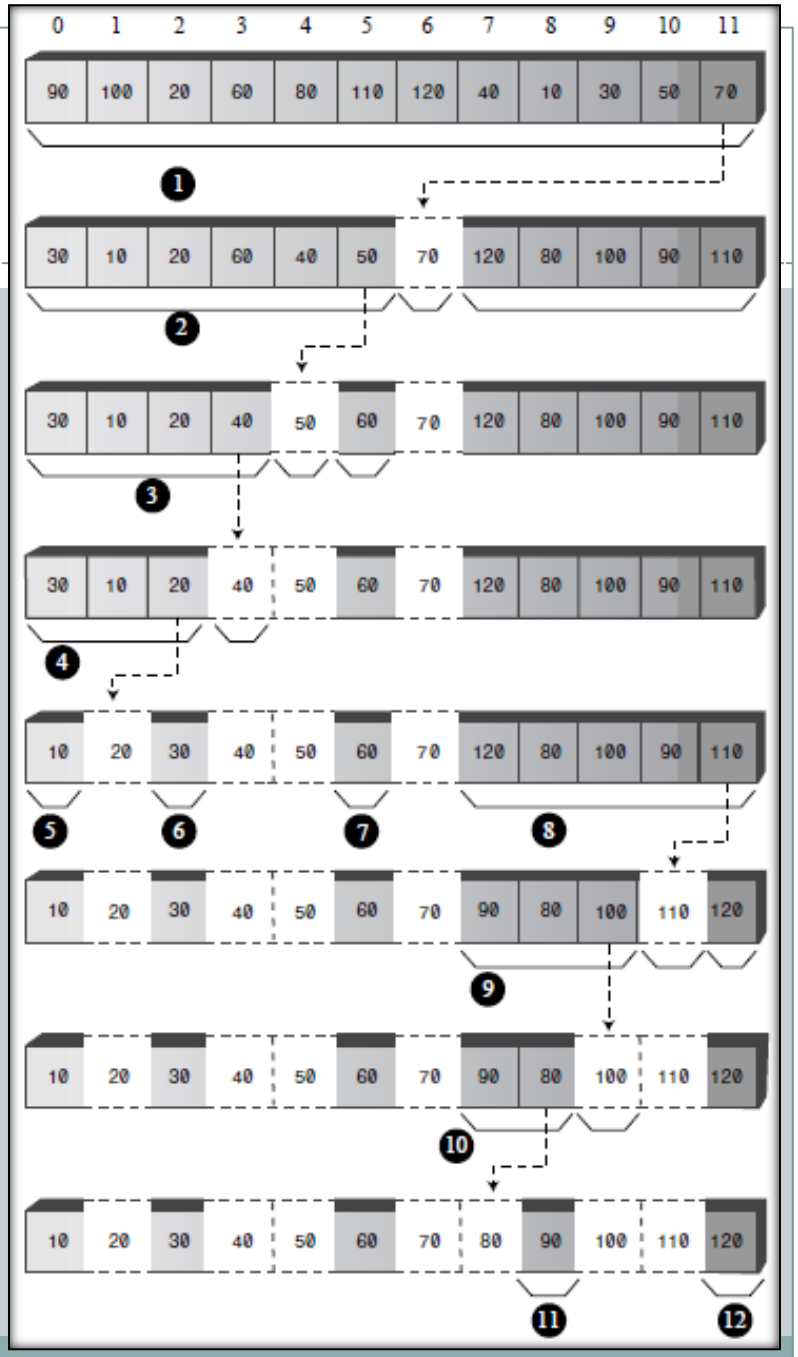


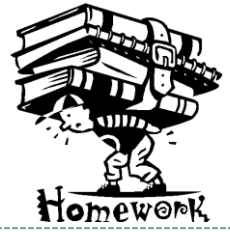
Quicksort: Pseudo Code



```
recQuickSort(left, right) {
  if Array has only element
      return; //already sorted
  else { //size is 2 or larger
    pivot is the right element of Array //rightmost item
   //partition range
   partition <- PartitionIt(left, right, pivot);
   recQuickSort(left, partition-1); //sort left side
   recQuickSort(partition+1, right); //sort right side
  }
} // end recQuickSort()
```

Quicksort Process





Homework 15 submit to:

fe.assignment@gmail.com

Fri., 18-Dec-2015 @ 15:00

Write a program:

1. to partition Array;
2. Use recursion to create Quicksort function.

Read book of **Robert Lafore**, page: 205– 279 for next lecture

Late submission: the score will be minus 10% for every hour

To be continued…