COMP 110-001
Introduction to Sorting

Yi Hong

June 12, 2015

Today

- Introduction to Sorting
  - Bubble sort
  - Selection sort
  - Merge sort

  - You should understand the idea behind bubble sort & selection sort
  - You should be able to **understand** the code given in slides (and **know how to use the code in similar problems by making slight modifications**).

Bubble Sort (or Sinking Sort)

- Basic idea (Wikipedia)
  - Start from the beginning of the list
  - Compare every adjacent pair, swap their positions if they are not in the right order
  - After each iteration, one less element (the last one) is needed to be compared until there is no more elements left to be compared

Animation from Wikipedia:

6 5 3 1 8 7 2 4

Bubble Sort

- **Step-by-step example of** "5 1 4 2 8“
- **First Pass:**
  (**5 1** 4 2 8) (**1 5** 4 2 8), Compares the first two elements, and swaps because 5 > 1.
  (1 **5 4** 2 8) (1 **4 5** 2 8), Swaps because 5 > 4
  (1 4 **5 2** 8) (1 4 **2 5** 8), Swaps because 5 > 2
  (1 4 2 **5 8**) (1 4 2 **5 8**), In order (8 > 5), no need to swap
- **Second Pass:**
  (**1 4** 2 5 8) (**1 4** 2 5 8)
  (1 **4 2** 5 8) (1 **2 4** 5 8), Swap since 4 > 2
  (1 2 **4 5** 8) (1 2 **4 5** 8)
  (1 2 4 **5 8**) (1 2 4 **5 8**)
  Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.
- **Third Pass:**
  (**1 2** 4 5 8) (**1 2** 4 5 8) No swap
  (1 **2 4** 5 8) (1 **2 4** 5 8) No swap
  (1 2 **4 5** 8) (1 2 **4 5** 8) No swap
  (1 2 4 **5 8**) (1 2 4 **5 8**) No swap, done.

Bubble Sort (I)

```
public static void bubbleSort(int [] data)
{
    for(int k = 0; k < data.length-1; k++)
    {
        for(int i = 0; i < data.length - 1 - k; i++)
        {
            if(data[i] > data[i+1])
            {
                // swap data[i] and data[i+1]
                int temp = data[i];
                data[i] = data[i+1];
                data[i+1] = temp;
            }
        }
    }
}
```

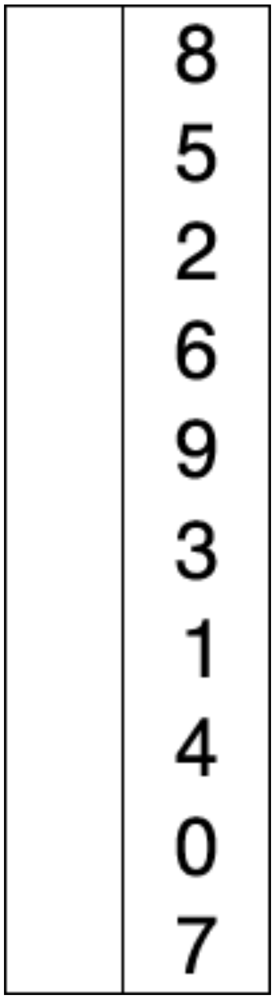
Bubble Sort (II)

```
public static void bubbleSort(int [] data)
{
    for(int k = 0; k < data.length-1; k++)
    {
        boolean bSwap = false;
        for(int i = 0; i < data.length - 1 - k; i++)
        {
            if(data[i] > data[i+1])
            {
                // swap data[i] and data[i+1]
                int temp = data[i];
                data[i] = data[i+1];
                data[i+1] = temp;
                bSwap = true;
            }
        }
        if(!bSwap) break;
    }
}
```

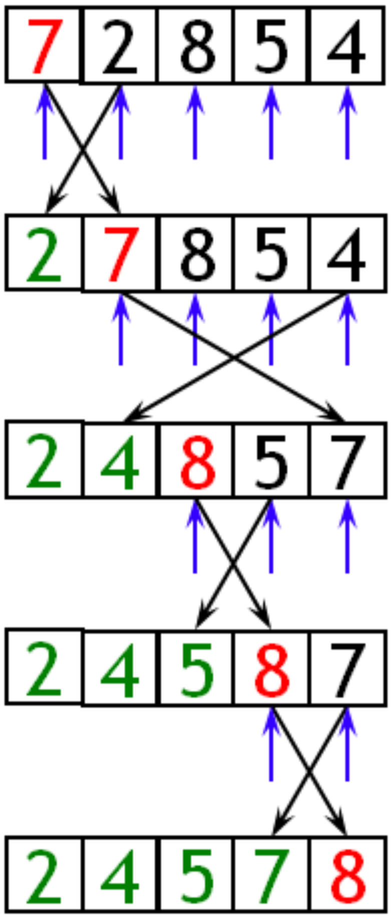
Selection Sort

- Given an array of length n, each time select the smallest one among the rest elements:
  - Search elements 0 through n-1 and select the smallest
    - Swap it with the element at location 0
  - Search elements 1 through n-1 and select the smallest
    - Swap it with the element at location 1
  - Search elements 2 through n-1 and select the smallest
    - Swap it with the element at location 2
  - Search elements 3 through n-1 and select the smallest
    - Swap it with the element at location 3
  - Continue until there's no element left

Animation from [Wikipedia](https://en.wikipedia.org):



An Example of Selection Sort



- Step by step example: 7 2 8 5
  - Iteration 1: found 2, swap it with 7
  - Iteration 2: found 4, swap it with 7
  - Iteration 3: found 5, swap it with 8
  - Iteration 4: found 7, swap it with 7
- The selection sort might swap an array element with itself--this is harmless, and not worth checking

Selection Sort

```
public static void selectionSort(int[] anArray)
{
    for(int i = 0; i < anArray.length - 1; i++)
    {
        int iSmallest = getIndexOfSmallest(i, anArray);
        swap(i, iSmallest, anArray);
    }
}

public static int getIndexOfSmallest(int startIndex, int[] a)
{
    int min = a[startIndex];
    int indexOfMin = startIndex;
    for(int index = startIndex + 1; index < a.length; index++)
    {
        if(a[index] < min)
        {
            min = a[index];
            indexOfMin = index;
        }
    }
    return indexOfMin;
}
```

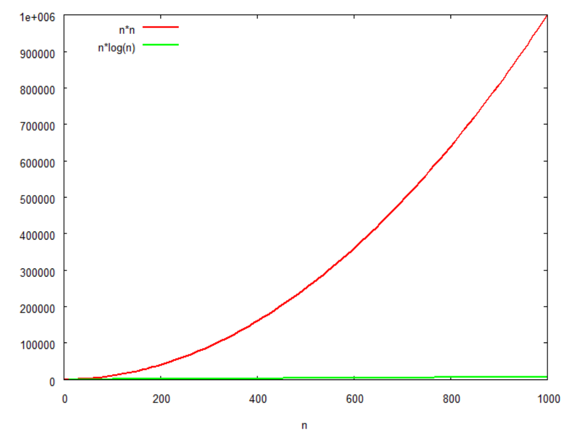
```
public static void swap(int i, int j, int[] a)
{
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

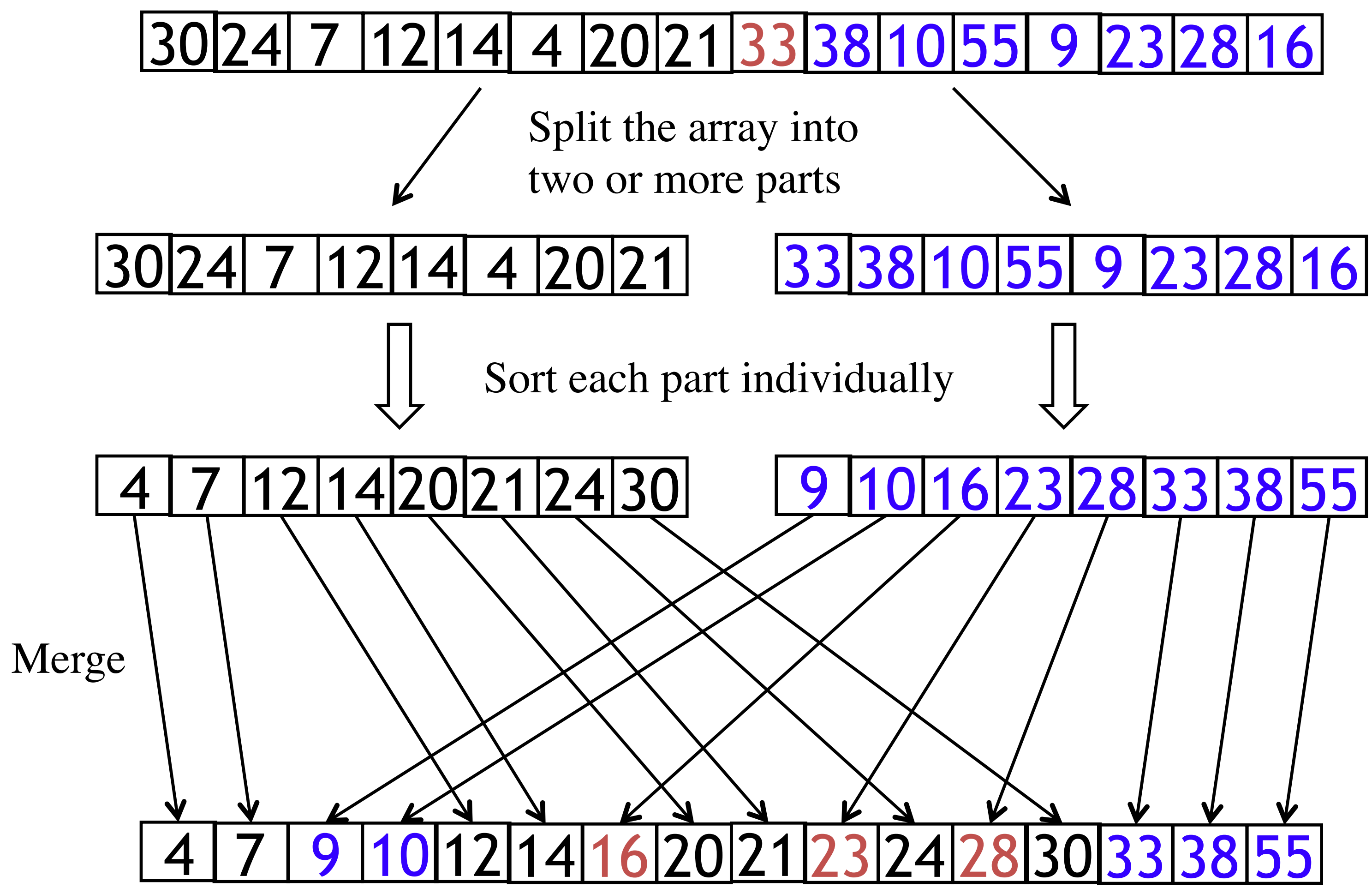
Merge Sort

- Bubble Sort and Selection Sort:
  - Intuitive and easy to implement
  - Help build basic abstract sorting concepts
  - Requires **~n^2 * c** operations in worst case
    - n : number of items to sort
    - c : some constant factor
  - Not commonly used in practice
- Two commonly used sorting algorithms in practice:
  - Quick Sort & Merge Sort

Merge Sort

- <u>Strategy:</u> Recursively split the list in half and merge the two returned segments
- Java's built-in sort function is a variant of merge sort: **Collections.sort(..);**
  - **Quick sort: Arrays.sort(..);**
- ~ **n*log(n) * c** operations in worst case
  - Check the difference between **n*log(n)** and **n^2** when n is large

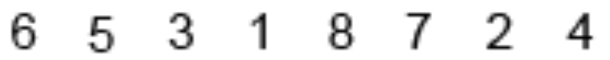




30 24 7 12 14 4 20 21 33 38 10 55 9 23 28 16
Split the array into two or more parts
30 24 7 12 14 4 20 21
33 38 10 55 9 23 28 16
Sort each part individually
4 7 12 14 20 21 24 30
9 10 16 23 28 33 38 55
Merge
4 7 9 10 12 14 16 20 21 23 24 28 30 33 38 55

Merge Sort

- Animation from [Wikipedia](https://en.wikipedia.org/wiki/Merge_sort):



Merge Sort

- Not easy to implement Merge sort correctly
- No Java code here (beyond the level of COMP110)
- Just understand the high-level idea

Next Class

- Review of final exam