Data Structures (Spring 2020)

Sorting Algorithms – 2 (Final Lab)

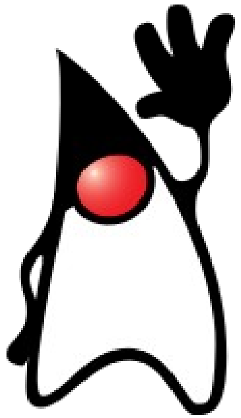
2020.06.12

Seoul National University

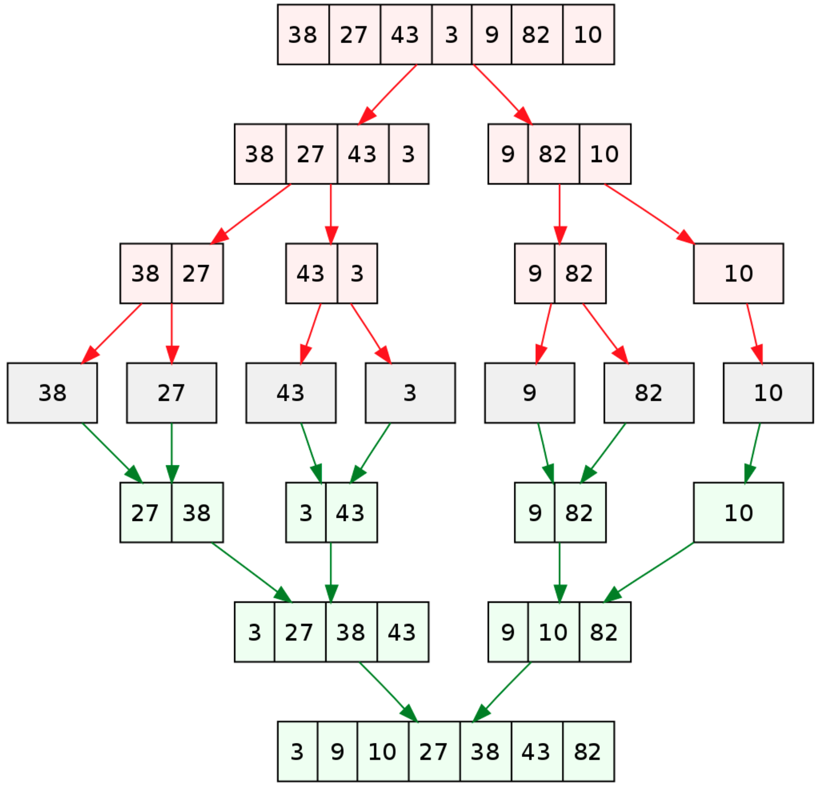
Database Systems Lab

Today's Lab

- Sorting Algorithm
  - Merge Sort
  - Quick Sort



Merge Sort



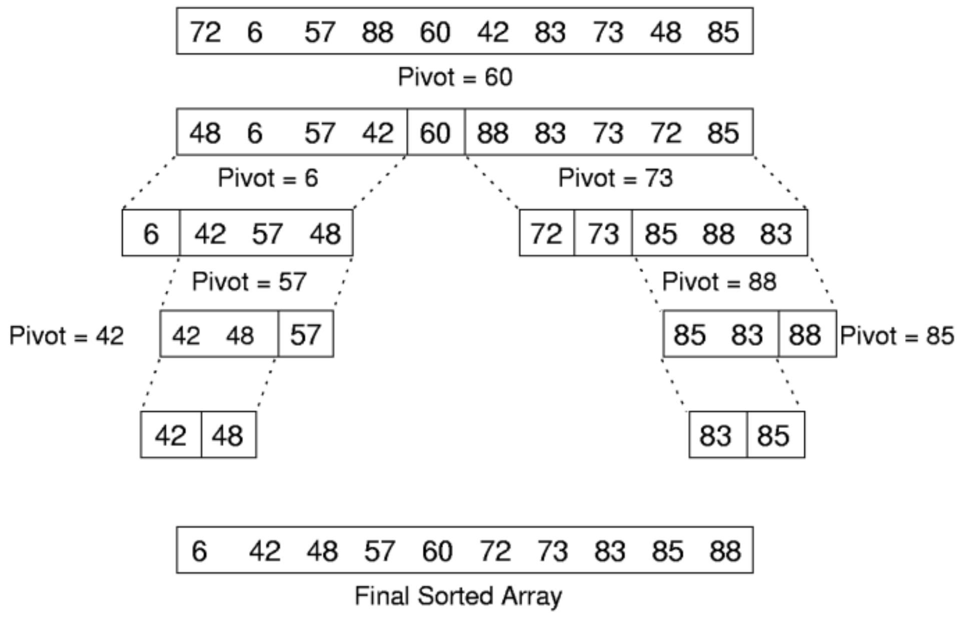
Algorithm 9 (Mergesort)

```
Marge(A,B)            // Both A and B are in sorted order
    for(i=j=k=0; i < |A| and j < |B| ;) {
        if (A[i] < B[j]) temp[k++] = A[i++];
        else temp[k++] = B[j++];
    }
    // Inv:  $i \geq |A|$ or $j \geq |B|$
    if (i < |A|)
        while(i < |A|) temp[k++] = A[i++];
    else
        while(j < |B|) temp[k++] = B[j++];
    return temp;

Mergesort(A)
    if (|A|==1) return A;
    B = Mergesort(A[0 ..  (N/2-1)]);
    C = Mergesort(A[N/2 ..  N-1]);
    return Merge(B,C);
```

"Merge sort", https://en.wikipedia.org/wiki/Merge_sort
From: Bongi Moon, "Lecture Notes on Data Structures: Sorting"

Quick Sort



Algorithm 7 (Quicksort)

```
Quicksort(A,left,right)                      // input:  A[left : right]
   if (left >= right) return;
   pivot = A[left];                          // the first one as a pivot
   for(i=left,j=right+1; i < j ;) {
      while(i < right and A[++i] < pivot);
      while(j > left and A[--j] > pivot);
      // Inv:  (A[i] ≥ pivot or i = right)
      //             and (A[j] ≤ pivot or j = left)
      // Inv:  if (i < j) then A[i] ≥ pivot and A[j] ≤ pivot
      if (i < j) swap(A[i],A[j]);
   }
   // Inv:  left ≤ j and A[j] ≤ A[left] = pivot
   swap(A[left],A[j]);        // place the pivot between partitions
   // Inv:  A[left:j-1] ≤ A[j] ≤ A[j+1:right]
   Quicksort(A,left,j-1);
   Quicksort(A,j+1,right);
```

From: Bongi Moon, "Lecture Notes on Data Structures: Sorting"

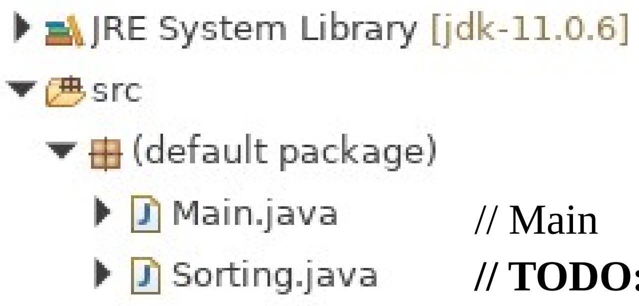
Sorting spec

- **public static int[] MergeSort(int[] data);**
  - int[] data: an input array
  - Return output sorted array.
  - Perform merge sort with input argument (array).
  - Split all elements into buckets and merge two buckets incrementally.

- **public static void QuickSort(int[] data);**
  - int[] data: an input array and an output sorted array
  - Perform quick sort with input argument (array).
  - Left side of pivot is smaller, right side of pivot is larger.

Exercises

- Fill the blank of codes
  - Update your code in Sorting.java ("// TODO: " section)
  - Write MergeSort(), QuickSort() method



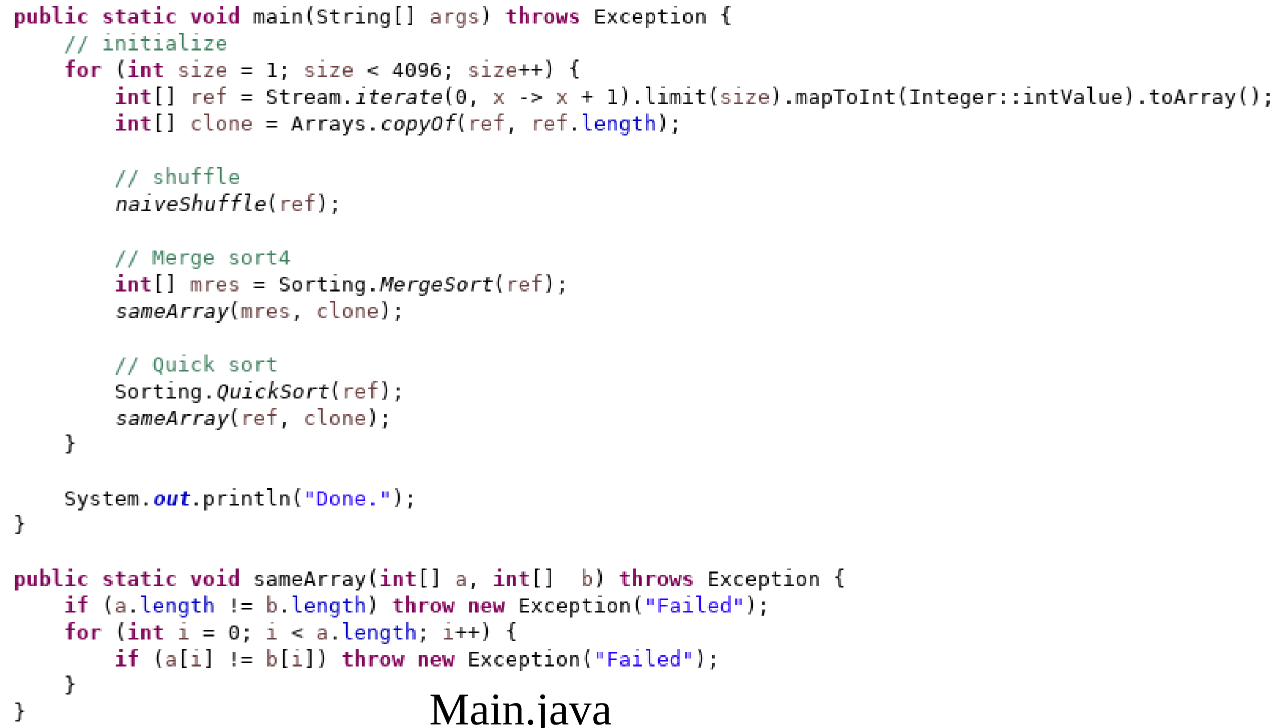
// Main

// TODO: Sorting

Project Structure

```
$ java Main
Done.
```

Result



```java
public static void main(String[] args) throws Exception {
    // initialize
    for (int size = 1; size < 4096; size++) {
        int[] ref = Stream.iterate(0, x -> x + 1).limit(size).mapToInt(Integer::intValue).toArray();
        int[] clone = Arrays.copyOf(ref, ref.length);

        // shuffle
        naiveShuffle(ref);

        // Merge sort4
        int[] mres = Sorting.MergeSort(ref);
        sameArray(mres, clone);

        // Quick sort
        Sorting.QuickSort(ref);
        sameArray(ref, clone);
    }

    System.out.println("Done.");
}

public static void sameArray(int[] a, int[]  b) throws Exception {
    if (a.length != b.length) throw new Exception("Failed");
    for (int i = 0; i < a.length; i++) {
        if (a[i] != b[i]) throw new Exception("Failed");
    }
}
```

Main.java