MERGE SORT

COP 3502

Introduction

- Earlier we covered Insertion Sort, Bubble Sort, and Selection Sort.
  - In these algorithms we end up making a significant number of possible comparisons and swaps between elements.
  - All of these have a worst and average case performance of $O(n^2)$.
- Is there a more clever, quicker way to sort numbers that does not require looking at most possible pairs of numbers?
- Today we will talk about MergeSort that uses recursion and a clever idea in sorting two separately sorted arrays.

The Merge

- The merging of two sorted lists is a tool we can use in Merge Sort.
- Say you are given 2 arrays, each of which is already sorted.
  - Now your job is to efficiently combine the 2 arrays into 1 larger one which contains all of the values of the 2 smaller arrays in sorted order.

The Merge

- The essential idea is this:

Array 1:

2	7	16	44	55	89
minA					

Array 2:

1	6	9	13	15	49
~~minB~~	minB				

Merged:

1											

1) Keep track of the smallest value in each array that hasn't been placed in order in the larger array yet.
2) Compare these two smallest values from each array. Place the smallest of the two values in the next location in the larger array.
3) Adjust the smallest value for the appropriate array.
4) Continue this process until all values have been placed in the large array.

- What does this remind you of? We talked about an algorithm that combines 2 sorted lists of names…
  - Sorted List Matching Problem

Example on the Board

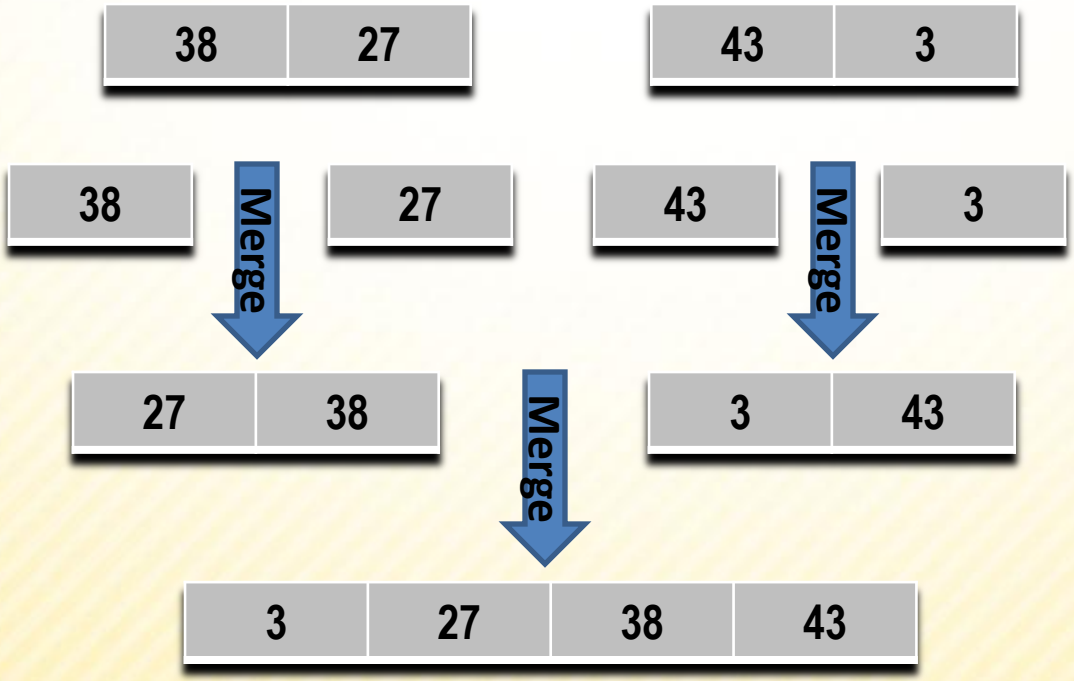
- Complete last example of merge on the board.

Merge Sort

- How can we use the Merge function to sort an entire array, since we we're merging special arrays where the 1st and 2nd halves were already sorted?

- The main idea:
  1) Sort the first half of the array, using merge sort.
  2) Sort the second half of the array, using merge sort.
  3) Now, we do have a situation to use the Merge algorithm. Simply merge the first half of the array with the second half.
- So all of the actual sorting gets done in the Merge method.
- Let's do an example to demonstrate this.

```
void MergeSort(int values[], int start, int end) {
    int mid;
    if (start < end) {    // Check if more than 1 element
        mid = (start+end)/2;
        MergeSort(values, start, mid);  // Sort 1st half
        MergeSort(values, mid+1, end);  // Sort 2nd half
        Merge(values, start, mid+1, end);
    }
}
```



Merge Sort Analysis

- Shown on the board

Practice Problem

Show contents of the array after each merge occurs in the process of Merge-Sorting the array below:

Initial	3	6	8	1	7	4	5	2
Sorted