Merge Sort

Melissa Thai, Ian Ludden, Abigail Anderson, Tyler Whitehouse

Sources:

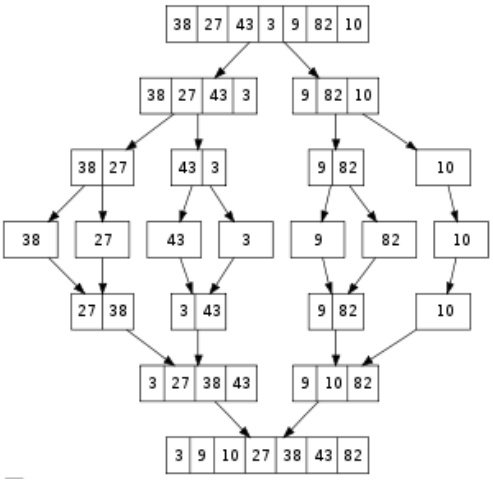
Big Java: Early Objects

http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Merge_sort.html

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/mergeSort.htm

---

Merge sort is a divide and conquer algorithm that sorts an array by cutting the array in half, recursively sorting each half, and then merging the sorted halves. The diagram below shows a visual representation of how the merge sort algorithm works:



The following are the general steps to use merge sort to sort an array A[x ... y]:

1. Divide Step

   If a given array A has zero or one element, simply return; it is already sorted. Otherwise, split A into two sub-arrays A[x … m] and A[m+1 … y] (m is the halfway point of A[x … y]), each containing about half of the elements of A.
2. Conquer Step

   Conquer by recursively sorting the two sub-arrays A[x … m] and A[m+1 … y].
3. Combine Step
   a. Combine the elements back into A[x … y] by defining a method that merges the two sub-arrays A[x … m] and A[m+1 … y] into a sorted sequence.

In sorting n objects, merge sort has a best and worst-case performance of O(n log n), which makes it an extremely efficient algorithm to use for sorting large amounts of data.

```java
public class MergeSorter {
	/**
	 * Sorts an array, using merge sort.
	 *
	 * @param a
	 *            the array to sort
	 */
	public static void sort(int[] a) {
		if (a.length <= 1) {
			return;
		}
		int[] first = new int[a.length / 2];
		int[] second = new int[a.length - first.length];
		// Copy the first half of a into first, the second half into second
		for (int i = 0; i < first.length; i++) {
			first[i] = a[i];
		}
		for (int i = 0; i < second.length; i++) {
			second[i] = a[first.length + i];
		}
		sort(first);
		sort(second);
		merge(first, second, a);
	}

	/**
	 * Merges two sorted arrays into an array
	 *
	 * @param first
	 *            the first sorted array
	 * @param second
	 *            the second sorted array
	 * @param a
	 *            the array into which to merge first and second
	 */
	private static void merge(int[] first, int[] second, int[] a) {
		int iFirst = 0; // Next element to consider in the first array
		int iSecond = 0; // Next element to consider in the second array
		int j = 0; // Next open position in a

		// As long as neither iFirst nor iSecond is past the end, move
		// the smaller element into a
		while (iFirst < first.length && iSecond < second.length) {
			if (first[iFirst] < second[iSecond]) {
				a[j] = first[iFirst];
				iFirst++;
			} else {
				a[j] = second[iSecond];
				iSecond++;
			}
			j++;
		}

		// Note that only one of the two loops below copies entries
		// Copy any remaining entries of the first array
		while (iFirst < first.length) {
			a[j] = first[iFirst];
			iFirst++;
			j++;
		}
		// Copy any remaining entries of the second half
		while (iSecond < second.length) {
			a[j] = second[iSecond];
			iSecond++;
			j++;
		}
	}
}
```