Leonardo Framba

Region Merge

Summary of Algorithm:

Region Sort is an Algorithm based on Merge Sort that aims to sort an inputted array in the quickest time possible. Region Sort employs a “divide and conquer” approach to separate the unsorted array and remerge it into a sorted one. Region Sort initially divides the unsorted array into a list of sorted regions. These regions can vary in size depending on the data. In the best-case scenario, the imputed array is already sorted and so there is only one region. In the worst-case scenario, the array is sorted in descending order in which case each element of the array will be separated into its own region. Once the array is separated into regions the algorithm continually merges two regions at a time until only one sorted region remains.

Methods:

Region Merge uses three methods(ignoring sort).

splitRegions(double inputArr)- This method separates the array into a list of sorted regions. We iterate through the array while keeping track of our starting index. Whenever we encounter an element that is out of order we use it as our ending index of the region. Then we update our starting index and continue through the array. The final region is added after the iteration is complete. The sorted regions are stored in a list of double arrays.

sortRegions(ArrayList<double> sortedRegions)- This method takes the list of sorted regions and continuously merges two regions together until only one region remains. Two regions are removed from the start of the sorted region list, merged, and then placed at the end of the list. This effectively replaces two regions with one decrementing the list size until one region is obtained.

mergeRegions(double[] firstArr, double[] secondArr)- standard merge sort method that merges two sorted regions into a single sorted region. Returns the merged region.

Comparison Vs Merge:

Although Region Sort is supposed to be an improvement over Merge, in my tests it was consistently slower by a small margin. This leads me to believe that Merge is more efficient when sorting large data sets of completely random data. Region sort shines when the data set has large regions of already sorted data. When the input array is already sorted Region is quicker than Merge running at a best case of O(n) vs Merge’s O(nlogn).