CS 260
Mergesort

Mergesort

Mergesort is the simplest of the recursive sorts covered in this module. It is a stable sort, so items that share the same key retain the order they had before the sorting. So, if the data needs to be sorted on multiple keys, this is the one to use.

The big problem with mergesort is that it uses twice as much space as any of the other recursive sorts, since it requires a second array used when merging the two arrays and data is copied multiple times from one array to another. It works well when sorting data that is accessed sequentially such as on disk drives since array accesses are to nearby locations instead of distributed as in quicksort or heapsort.

As with the other recursive sorts described here, its performance is order of N log N.

The basic concept is, given two arrays which are already sorted, they can be merged into a single sorted array. To have two arrays that are sorted, sort each of them recursively using merge sort. That is, divide the array into smaller and smaller halves until each half is of length one, then merge them back together.

Example

Here is a simple example of using merge sort. The goal this array sorted into ascending order from left to right. Heavy borders around cells to indicates where the array is split into subarrays. Each row represents one iteration of the sort.

3	6	5	1	Starting
3	6	5	1	Divide in half
3	6	5	1	Divide halves
3	6	5	1	Merge left
3	6	1	5	Merge Right
1	3	5	6	Merge All

Algorithm

Merge sort is normally done with two functions. One is the recursive sort itself that repeatedly divides the array into subarrays. The other is a merge function that joins them back together.

Merge

```
Pass in two arrays to merge
Create new destination array
index1 = start of first array, index2 = start of second array
While index1 < length1 and index2 < length2
    If array1[index1] < array2[index2]
        Copy array1[index1] to destination array
        Index1 += 1
    Else
        Copy array2[index2] to destination array
        Index2 += 1

Copy remaining items from the array that is not done to destination array

Return the destination array
```

RecMergeSort

```
Pass in array
Split in half
Recursively sort the first half
Recursively sort the second half

newArray = Merge first half and second half

copy newArray to parameter
```