Part 1: When you write an email to your family, the email is sent over the Internet. Before the message is sent, it is broken into many data packets. These data packets are numbered consecutively, but they are out of order while transferred over the network. The packets need to be sequenced when they reach the destination computer so your family can read your message. Discuss how you would sort these packets with two kinds of sort algorithms.

**Assignment Expectations**

1. Discuss and explain clearly why you chose these two kinds of algorithm.
2. Use numbers to demonstrate how these algorithms work. Hint: Create a series of numbers to represent the data packets.

Note: it is okay for you to draw on paper, but you must you can take a clear picture and paste it into your Word document.

Part 2: Choose one sort algorithm from the ones covered in this module and write a program to sort students’ GPA in descending order. After you are done, send the original Java code along with screenshots of the result.

The two sorting algorithms I would choose are the quickSort() and the mergeSort(). They are both relative fast sorting alorithms with runtimes of O(nlog(n)). Quick sort has a worst case that is O(n^2), but since the emails send finite n number of packets the worst case scenarios will seldom be met.

QuickSort()

First, grab the final element in the array 🡪value of 2 (Pivot)

Place a wall at the beginning of the list or array of numbers.

Next, look at

1. The lowest index (right of the wall) or the left most element (the current element)

Now, we compare each element against the pivot until we find a value less than the pivot.

We then swap the first element in the array with this value and move the wall right one element.

After iterating through all the values and coming to the pivot, we insert the pivot at the wall by swapping the value of the element at the right of the wall with the pivot and increment the wall by one.

This is repeated, recursively, until the list is sorted.

Note: The algorithm will run at its worst case scenario, O(n^2), if the beginning element and its pivot are either the lowest or highest value.   
**To reduce the risk of this worst case scenario, we can take the median of three elements and use this value as our pivot.**

Example:

{3,4,5,1,2}

{WALL,3,4,5,1, Pivot22}

Go through array and see 1 is less than the pivot.

{WALL,1,4,5,3, Pivot22}

Increment WALL

{1,WALL,4,5,3, Pivot2}

Since all values to the right of the Pivot are greater than the Pivot, we swap the Pivot and the value right of the WALL

{1,WALL,2,5,3, Pivot4}

Increment Wall

{1,2,WALL5,3, Pivot4} REPEAT earlier steps

{1,2, 3, WALL5, Pivot4} REPEAT earlier steps

{1,2,3,4,5}

MergeSort()

Step one, recursively split the list in half until there is only one value in each sub-list. (If list is of odd values, just pick a list to put the extra element)

Next, merge pairs of sub-lists so that the sub arrays are ordered. Start by looking at the first element in each array, making a comparison, and placing the lesser value in the merged array.

Continue this process until all the values are in sorted order and in one list.

Example:

{3,4,5,1,2}

{3,4,5} {1,2}

{3} {4,5} {1} {2}

{3} {4} {5} {1} {2}

Now merge

{3,4} {1,5} {2}

Merge again

{1,3,4,5} {2}

Merge one last time

{1,2,3,4,5}

www.houzz.com