**HW11b: Sorting and Searching part 2**

1. The sort method in our MergeSort class created new sub-arrays for the left and right parts of the array. Creating these arrays for each recursive call takes time and creates an unnecessary inefficiency in the program. If we use a recursive problem transformation to introduce a new helper method this inefficiency can be removed. The class below shows how the sort method might be transformed by adding parameters for start and end, which indicate the portion of the array to be sorted. It also gives the signature of a transformed merge method. Assuming the new merge method works, give an implementation of the new sort method.

**public** **class** MergeSortWithHelper {

/\*\*

 \* Merge sort the provided array.

 \*

 \* **@param** arr the array to be sorted.

 \* **@return** the sorted array.

 \*/

**public** **static** **int**[] sort(**int**[] arr) {

**return** *sort*(arr, 0, arr.length);

}

/\*

 \* Sort the contents of arr from index start, up to

 \* but not including end.

 \*/

**private** **static** **int**[] sort(**int**[] arr, **int** start, **int** end) {

// Give an implementation of this method.

}

/\*

 \* Merge the contents of the array from start up to but not

 \* including mid with the contents from mid up to but not

 \* including end. Assume that [start…mid) and [mid…end) are

 \* in sorted order. Return arr with the indicated sections merged.

 \*/

**private** **static** **int**[] merge(**int**[] arr, **int** start, **int** mid, **int** end){

// Assume this method works.

 }

}

2. The linear search algorithm is defined as:

 search(arr, x): return the index at which x is found in arr

or -1 if x is not contained in arr.

If we add a parameter indicating the index at which to start the search it becomes possible to give a recursive definition for linear search.

 search(arr, x, start): return the index at which x is found in arr

or -1 if x is not contained in the portion of arr from index start to the end of the array.

Give a recursive definition of linear search using this second definition of the search method. Clearly state the base case(s) and the recursive case(s). Do not implement your definition in Java.

3. In class we developed the following recursive definition for binary search:

search(arr, x, start, end) // start and end are inclusive

 Let mid = (start+end) / 2

Base Cases: start>end return -1

arr[mid] == x return mid

Recursive Cases: arr[mid] > x return search(arr, x, start, mid-1)

 arr[mid] < x return search(arr, x, mid+1, end)

Consider the following sorted list of integers:

 4 6 9 10 12 15 21 28 33 45 49 50 58

The first value in this list that would be inspected by our binary search algorithm is 21. Give the full sequence of values that would be inspected by the above binary search algorithm when searching for each of the following values:

 a. 49

b. 7

4. Consider the following unsorted list of integers:

45 4 33 6 28 10 50 9 12 58 15 21 28

a. Give the full sequence of values that would be inspected by our binary search algorithm if it was used to search this list for the value 9.

b. Explain why binary search requires that the array be in sorted order.

5. Modify the binarySearch and binarySearchHelper methods in the Search class so that they can search an array of Student objects. The Search and Student classes are available on the “Homework assignments” web page. Copy and paste the modified methods as your answer to this question.

6. Modify the compareTo method for the Student class so that Students will be sorted in order by graduation year and alphabetically within graduation year. Thus, all students with the earliest graduation year are listed before any other students, and any group of students with the same graduation year are listed consecutively, alphabetically by name. Copy and paste your modified compareTo method as your solution to this question.

7. Define a class that implements Comparator. The new class will enable sorting a list of Student objects into order by increasing graduation year and by increasing GPA within graduation years. Two students graduating in the same year with the same GPA should be ordered alphabetically. Paste your new class here.

Bonus: Give an implementation of the transformed merge method as defined in problem #1.