In this assignment you will implement several array algorithms including mergesort pages 630–640 (3rd ed: pp. 614–624). These algorithms will be used for a web page spell checker.

1 Concerning Merge Sort and Other Sorted Array Algorithms

All the algorithms are to be implemented in a class SortUtils that is created with a comparator that is to be used by all the algorithms.

The textbook described the mergesort algorithm. One of the awkward aspects of mergesort is that it is tricky to merge two segments of an array in place. You will implement mergesort using several helper methods which use explicit temporary areas.

merge(lo,mid,hi,in,out) This method should take two (presumably) sorted array segments of in, and merge them into the same area of out. The two array segments are the elements in the range [lo,mid) and [mid,hi) respectively. The merged elements will go into array out in the range [lo,hi) which of course is of size equal to the sum of the sizes of the input segments. The code can rely on the segments being sorted (according to the comparator), the arrays being non-null and big enough and the output array being different than the input array.

merge(1,3,5, \{5,2,9,6,7,0\}, A, B, C, D, E, F)  
Afterwards: in unchanged and out = [\{1,2,6,7,9\}]F

mergeSortKeep(lo,hi,data,temp) This method should sort the array segment [lo,hi) in the data array and write the results back to the same array. The temporary array temp (which will be valid for the same range of elements) can be used for extra space. Again you can rely on the arrays not being null, being big enough and being different.

mergeSortKeep(1,4, \{5,9,2,6,7,0\}, A, B, C, D, E, F)  
Afterwards: data = \{5,2,6,9,7\}

mergeSortMove(lo,hi,in,out) This method should sort the array segment [lo,hi) in the in array and write the results to the out array. You can modify the elements in the input array in during the process. Again you can rely on the arrays not being null, being big enough and being different.

mergeSortMove(1,4, \{5,9,2,6,7,0\}, A, B, C, D, E, F)  
Afterwards: out = [\{1,2,6,9\}]E

The two mergesort helper methods (mergeSortKeep and mergeSortMove) will usefully call each other (on smaller segments!) in recursive calls. It is required that you avoid excessive copying for full credit. Only in the base case (single element) are you allowed to simply copy elements from one array to another. Otherwise, all data movement must be accomplished as a side-effect during some useful task (e.g. merging).

Another algorithm you will implement is set difference (A – B). This removes all elements from one sorted segment that are in the second sorted segment. Since the algorithm doesn’t require temporary storage, the client can have the output array be the same as the input array.

difference(lo1,hi1,lo2,hi2,in,rem,out) Write to out[lo1,...] all the elements from the sorted array segment in[lo1,hi1) that do not occur in the sorted array segment rem[lo2,hi2). Return the next available index in the output array segment.
difference(1,6,2,5, 0 1 2 3 4 5, A B C D E F) = 5
Afterwards out = A 1 2 3 5 F

Since the resulting array segment may be smaller that the input segment, the method returns the (exclusive) upper bound. The last algorithm removes duplicates from sorted array segments:

uniq(lo,hi,in,out) Write to out[lo,...] all the “unique” elements from the sorted array segment in[lo,hi). Return the next available index in the output array segment.

uniq(1,5, 0 0 1 1 2 2, A B C D E F) = 4 Afterwards out = A 0 1 2 E F

Since the array is sorted, duplicate elements will be consecutive. As with difference, the input and output arrays can be the same.

2 Concerning the Web Spell-Checker

The algorithms will be used in a spell-checker that downloads a web page, getting at all the words and then comparing them again a dictionary of words. The downloading and parsing of HTML will be handled for you. The constructor reads in the dictionary. The other public method checks an XML element:

check(Element e) Using a private helper method, the words in the element are placed into an array (using an ArrayList temporarily as does the constructor), which can then be sorted, uniq’ed and differenced with the dictionary. A list of misspelled words is returned.

You should find words inside strings inside the XML at any depth except inside <script> and <style> elements. We recommend you use replaceAll to remove punctuation characters other than the apostrophe. The program runs on URLs in the “Program Arguments” section of the run configuration.

3 What you need to do

You need to complete the SortUtils and SpellCheck classes: the methods outlined in the previous sections. We will not be testing invariants in this assignment.

4 Files

The repository for this homework includes the following files:

src/Main.java Main program that delegates to SpellCheck.
src/Test{SortUtils,SpellCheck}.java JUnit test suites.
src/TestEfficiency.java Efficiency tests.
src/TestExhaustive.java Exhaustive test of merge sort.
src/edu/uwm/cs351/SpellCheck.java The spell-checker.
src/edu/uwm/cs351/utility/SortUtils.java The sorted array algorithms.
lib/dictionary.txt Dictionary of English words.

There is also a RandomTest of the five sort utility methods in the usual place.