Amritpal Saini

30039983

Questions:

1. Two different sorting algorithms are used for this assignment. Bubble Sort is used for inputs under 100 and for sorting the letters to ready them for the anagram check. Although the big O for bubble sort is N^2, it is safe to use for a small input because the difference is minimal. The longest word in the English language is shorter than 100 characters so it is safe to use for sorting the letters as well. Merge Sort is the algorithm used for inputs larger than 100 because it provides a quicker sort. Merge sort takes more resources than Bubble Sort thus is only used when necessary.

2. The running time for my program is $17 + 48N + 46N^2 + LN^2 + 42N^3 + L^2N + LN^3 + 38NLogN$ (base 2) + 9logN (base 2) which is $O(N^3L^2)$. Time Unit analysis from Appendix Tables 1-5.

3. By substituting in 2 for N, we can find the time units required based on the length of the word. $24 + 17 + 48(2) + 46(4) + L(4) + 42(8) + L^2(2) + L(8) + 38*(2log2)$ base 2 + (9log2) base 2= $718 + 12L + 2L^2$ time units

Appendix: Table 1: Analysis of Main function

Line #	Main	Cost	Times	Comments
1	StaticArray<String> listA = new StaticArray<String>(30000);	C1 =2	1	Initializing a StaticArray and assigning a variable
2	Scanner read = new Scanner(file);	C2 = 2	1	Initializing scanner and assigning a variable
3	listA.add(read.nextLine());	C3 = 2	1	Performing an operation and adding a value
4	while(read.hasNextLine()) {	C4 = 1	N+1	Operation, +1 for last check when it doesn't have any more lines
5	listA.add(read.nextLine());	C6 = 2	N	Operation
6	}			
7	if (listA.capacity() > 100) {	C7 = 2	1	Operation and comparison
8	listA.add(" ");	C8 = 1	1	Operation
9	listA.trimToSize();	C9 = 6	1	Operation
10	MergeSort(listA);	C10 = N + 38Nlog(N) + 9logN + 31	1	Call of method with Big(O) of N log(N)
11	}			
12	else {			
13	listA.trimToSize();	C11 = 6	1	Operation
14	bubbleSortString(listA);	C12 = 6N +18N^2-28	1	Call of Method with Big(O) of N^2
15	}			
16	StaticArray<SinglyLinkedList<String>> listB = generateListB(listA);	C13 = -4 + 37N +28N^2+6L +18L^2+LN^2 +42N^3+L^2 N+LN^3	1	Assign variable and call of method with time of 22 + 37N +28N^2+LN^2+43N^3+LN^3
17	BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile));	C14 = 2	1	Assigning a variable and operation
18	writer.write(toString(listB));	C14 = N	1	Call on method with Big(O) of N
19	writer.close();	C15=1	1	Operation
20	}			
	17 + 48N +46N^2+LN^2+42N^3+L^2N+LN^3 + 38NLogN + 9logN = O(N^3L^2)			

Table 2: Analysis of GenerateListB function

Line #	GenerateListB	Cost	Times	Comments
1-3	StaticArray<SinglyLinkedList<String>> listB = new StaticArray<SinglyLinkedList<String>>(listA.size()); StaticArray<StaticArray<String>> Middle = new StaticArray<StaticArray<String>>(listA.size());	C1 =4	1	Assigning a variable and operation
4	for(int i = 0; i < listA.capacity(); i++) {	C2 = 1 C3 = 1 C4 = 2	1 N + 1 N	Assigning variable Comparison, plus 1 for fail Arithmetic and assigning variable
5	String tmp = listA.get(i);	C5 = 2	N	Assigning variable and operation
6	StaticArray<String> letters = new StaticArray<String>();	C6 = 2	N	Assigning variable and operation
7	for(int j = 0; j < tmp.length(); j++) {	C7=1 C8=1 C9=2	N N(N+1) N^2	Assigning variable Comparison Arithmetic and comparison
8	letters.add(String.valueOf(tmp.charAt(j)));	C10 = L + 2	N^2	Operation
9	}			
10	letters.trimToSize();	C11 = 6	N	Operation
11	bubbleSortString(letters);	C12 = 6L +18L^2-28	N	Method that has Big(O) N^2
12	Middle.add(letters);	C13 = 1	N	Operation
13	}			
14	Middle.trimToSize();	C14 = 6	1	Operation
15	StaticArray<String> dne = new StaticArray<String>(); dne.add("abc");	C15 = 3	1	Assigning variable, Creating Static Array, Operation
16	for(int i = 0; i < Middle.capacity(); i++) {	C16 =1 C17=1 C18 =2	1 N+1 N	Assigning variable Comparison Arithmetic and comparison
17	SinglyLinkedList<String> linkedList = new SinglyLinkedList<String>();	C19 = 2	N	Assigning variable and operation
18	if(Middle.get(i).get(0) != "abc") {	C20 = 3	N	2 Operations and comparison
19	linkedList.addFirst(listA.get(i));	C21 =2	N	2 Operations
20	StaticArray<String> arr1 = Middle.get(i);	C22 = 9	N	Assigning variable,

	arr1.trimToSize();			3 operations
21	for(int j = 0; j < Middle.capacity(); j++) {	C23=1 C24=1 C25=2	N N(N+1) N^2	Assigning variable Comparison Arithmetic and comparison
22	boolean ana = false;	C26 = 2	N^2	Assigning variable and giving it a value
23	if(j != i && (Middle.size()> j) && (arr1.get(0) != "abc")) {	C27 = 5	N^2	Comparisons and operations
24	StaticArray<String> arr2 = Middle.get(j); arr2.trimToSize();	C28 = 9	N^2	Assigning variable, 3 operations
25	ana = false;	C29 = 1	N^2	Assigning variable
26	if (arr2.capacity() == arr1.capacity()) {	C30 = 3	N^2	Comparison and operation
27	for(int k = 0; k < arr1.capacity(); k++) {	C31=1 C32=1 C33=2	N^2 $N^2(N+1)$ N^3	Assigning variable Comparison Arithmetic and comparison
28	if(!(arr1.get(k).equals(arr2.get(k)))) {	C34=L+4	N^3	Operation and comparison
29	ana = false;	C35 = 1	N^3	Assignment
30	k = arr1.capacity();	C36 =1	N^3	Assignment
31	} else {			
32	ana = true;	C37 = 1	N^3	Assignment
33	}}}			
34	if(ana) {	C38 =1	N^2	Comparison
35	for(int l = 0; l < linkedList.size(); l++) {	C39=1 C40=1 C41=2	N^2 $N^2(N+1)$ N^3	Assigning variable Comparison Arithmetic and comparison
36	if((linkedList.get(l).equals(listA.get(j)))) {	C42 = 3	N^3	Comparison and Operations
37	Middle.set(i, dne); Middle.set(j, dne); ana = false;	C43 = 11	N^3	Operation and assignment
38	}}			
39	if(listA.get(j).compareTo(linkedList.last()) != 0 && ana) {	C44 = 4	N^3	Operation and comparison

40	linkedList.addLast(listA.get(j));
	Middle.set(i, dne);
	Middle.set(j, dne);
	ana = false;
		C45 = 13	N^3	Operation and assignment

41	}}}}}			
42	if(!linkedList.isEmpty()) { listB.add(linkedList);	C46 = 3	N	Operation, comparison, and operation
43	}}			

44	listB.trimToSize(); return listB;	C47 = 7	1	Operation and return
	$-4 + 37N + 28N^2+6L+18L^2+LN^2+42N^3+L^2N+LN^3 = O(N^3)$			

Table 3

Line #	MergeSort	Cost	Time	Comments
1	int n = arr.size() - 1;	C1 = 3	1	Assignment, operation, arithmetic
2	if (!(n < 2)) {	C2 = 2	1	Compare
3	int mid = n/2;	C3 = 3	1	Assignment, operation, arithmetic
4	StaticArray<String> arr1 = arr.subArray(0, mid); StaticArray<String> arr2 = arr.subArray(mid,n);	C4= N +16	1	Iteration, for loop, asssingmnet
5	MergeSort(arr1); MergeSort(arr2);	C5 =2	2*F(n/2)	Call of method
6	merge(arr1, arr2, arr);	C6 = 38N+7	LogN	From table 4
	$N + 38N*\log(N) + 9\log N + 31 = O(N\log N)$ base 2			

Table 4

Line #	merge	Cost	Time	Comment
1	int i = 0; int j = 0;	C1 = 4	1	Assignment
2	while(i+j < arr.capacity()) {	C2 = 3	N+1	Comparison and arithmetic
3	if ((j == arr2.capacity() -1) || (i < arr1.capacity() -1) && (arr1.get(i).compareTo(arr2.get(j)) < 0)) {	C3 = 11	N	Comparison, arithmetic, operation
4	arr.set(i+j, arr1.get(i)); i++;	C4 = 12	N	Operation, arithmetic
5	} else {			
6	arr.set(i+j, arr2.get(j)); j++;	C5=12	N	Operation, arithmetic
7	}}}			
	38N+7 = O(N)			

Table 5

Line #	bubbleSortString	Cost	Time	
1	int n = listA.capacity() - 1;	C1 = 4	1	Assignment, operation, and arithmetic
2	int k = n;	C2=2	1	Assignment
3	boolean sorted = false;	C3=2	1	Assignment
4	while((k > 0) && (sorted == false)) {	C5 = 2	N-1	Comparison and fail test
5	sorted = true;	C6=1	N-2	Assignment
6	for(int i = 0; i < k; i++) {	C7=1 C8=1 C9=2	N N(N-1) N^2-2	Assigning variable Comparison Arithmetic and comparison
7	if(listA.get(i).compareTo(listA.get(i+1)) > 0) {	C10 =5	N^2-2	Operation, arithmetic, comparison

8	String tmp = listA.get(i);
	listA.set(i, listA.get(i+1));
	listA.set(i+1,tmp);
	sorted = false;
		C11 = 10	N^2-2	Assignment, operation

9	}}			
10	k--;	C12= 2	N	Arithmetic and assignment
	$6N +18N^2-28 = O(N^2)$			

Sources:

Sousa, M., CPSC 319, 01 - 02 - Sorting Algs. - (2) Bubble (Part 1) -- 1 sld-pp, 2019

Sousa, M., CPSC 319, 01 - 04 - Sorting Algs. - (4) Merge-Sort -- 1 sld-pp, 2019

Rashid, M., CPSC 319, pages.cpsc.ucalgary.ca/~mdmamunur.rashid1/CPSC319-W19.html, 2019

Rashid, M., CPSC 319, http://pages.cpsc.ucalgary.ca/~mdmamunur.rashid1/CPSC319T/codes/StaticArray.java.html, 2019

Rashid, M., http://pages.cpsc.ucalgary.ca/~mdmamunur.rashid1/CPSC319T/codes/SinglyLinkedList.java.html, 2019

Rashid, M., CPSC 319, http://pages.cpsc.ucalgary.ca/~mdmamunur.rashid1/CPSC319T/codes/asgmt2.java.html, 2019