Einsendeaufgabe 3

Stefan Berger

1.

Show how the algorithm works with the array (2, 4, 11, 12, 7, 15, 19) . In particular show how the values of m, i and k are changing, which elements are compared and which elements are exchanged, see script p. 50 example 4.7.

m	i	A	A[i] > A[i + 1]	Vertauschen	k
5	1	$(2, 4, 11, 12, 7, 15)$	$2 > 4$	nein	
	2	$(2, 4, 11, 12, 7, 15)$	$4 > 11$	nein	
	3	$(2, 4, 11, 12, 7, 15)$	$11 > 12$	nein	
	4	$(2, 4, 11, 12, 7, 15)$	$12 > 7$	ja	4
	5	$(2, 4, 11, 7, 12, 15)$	$12 > 15$	nein	
3	1	$(2, 4, 11, 7, 12, 15)$	$2 > 4$	nein	
	2	$(2, 4, 11, 7, 12, 15)$	$4 > 11$	nein	
	3	$(2, 4, 11, 7, 12, 15)$	$11 > 7$	ja	3
2	1	$(2, 4, 7, 11, 12, 15)$	$2 > 4$	nein	
	2	$(2, 4, 7, 11, 12, 15)$	$4 > 7$	nein	

3.

Test your method with the file QuickSort.txt Copyright Tim Roughgarden, Stanford University. For that used the 2 static methods in the class called UtilitiesArraysFiles. You can also write your own methods if you wish. One method reads the numbers from the textfile in an array, and the other writes the numbers of the array in a textfile whose name is given as a parameter. Theses methods have the following headers: public static int[] readInArray(String filename) public static void writeArrayToFile(int [] ir, String filename). How many comparisons do you get? Does this number fit the theoretical result O(n2) (in fact n(n- 1)/2, see script p. 50) in the worst case or O(n) in the best case?

The implementation makes 49916606 comparisons when sorting the numbers in the file QuickSort.txt. That's within the theoretical numbers for the best case and the worst case.

In the best case, the algorithm needs n-1 comparisons, (9999 with the numbers 1 - 9999 presorted).
$B(n) = n - 1$
$B(n) \in \Omega(n)$

In the worst case, the algorithm needs $n(n - 1)/2$ comparisons (49995000 with the numbers 1 - 9999 sorted in descending order).
$W(n) = n(n - 1)/2$
$W(n) \in \mathcal{O}(n^2)$