Florida International University

COP-3530

PI# 5913959

Assignment-4 Due: April 13, 2017



**Problem #1**: a) Given the following list of numbers:

[90, 8, 7, 56, 123, 235, 9, 1, 653]

trace the execution for:

a.1) Selection Sort (only the first 5 steps).

Assume increasing sort.

Step 1: [**1**, 8, 7, 56, 123, 235, 9, 90, 653]

Step 2: [**1, 7**, 8, 56, 123, 235, 9, 90, 653]

Step 3: [**1, 7, 8**, 56, 123, 235, 9, 90, 653]

Step 4: [**1, 7, 8**, **9**, 123, 235, 56, 90, 653]

Step 5: [**1, 7, 8**, **9**, **56**, 235, 123, 90, 653]

a.2) QuickSort (pivot = the first element).

Step 1: [8, 7, 56, 9, 1, **90**, 123, 235, 653]

Step 2: [7, 1, **8**, 56, 9, **90**, **123**, 235, 653]

Step 3: [1, **7**, **8**, 9, **56**, **90**, **123**, **235**, 653]

Step 4: [**1, 7, 8, 9, 56, 90, 123, 235, 653**]

a.3) MergeSort.

Step 1: [90, 8, 7, 56, 123] [235, 9, 1, 653]

Step 2: [90, 8, 7] [56, 123] [235, 9] [1, 653]

Step 3: [90, 8] [7] [56] [123] [235] [9] [1] [653]

Step 4: [90][8] [7] [56] [123] [235] [9] [1] [653]

Step 5: [8, 90] [7] [56] [123] [235] [9] [1] [653]

Step 6: [7, 8, 90] [123, 56] [9, 235] [1, 653]

Step 7: [7, 8, 56, 90, 123] [1, 9, 235, 653]

Step 8: [1, 7, 8, 9, 56, 90, 123, 235, 653]

b) The processing time of an algorithm is described by the following recurrence equation (c is a positive constant):

T(n) = 3T(n/3) + 2cn; T(1) = 0

What is the running time complexity of this algorithm? Justify.

The equation has the form: T(N) = aT(N/b) + O(Nd); so we can use the Master Theorem to solve it.

a= 3, b=3, d=1 🡪 a = bd => T(n) = **O(n log n)**

c) You decided to improve insertion sort by using binary search to find the position p where the new insertion should take place.

c.1) What is the worst-case complexity of your improved insertion sort if you take account of only the comparisons made by the binary search? Justify.

The worst case for the number of comparisons will decrease from **O(n2)** in the normal insertion sort algorithm to **O(n log n)** for the improved algorithm. There are n elements to compare in the array and log n comparisons in the worst case using binary search.

c.2) What is the worst-case complexity of your improved insertion sort if only swaps/inversions of the data values are taken into account? Justify.

The worst case complexity will remain the same after our improvement, since the binary search will not change the number of swaps in the original algorithm. **O(n2)** time complexity since we still need n swaps for n elements.

d) What is the running time complexity of the QuickSort when all elements of the array have the same value? Justify.

If all elements are the same, no matter what pivot we picked, the resulting groups will be unbalanced since 1 group will always have the rest of the elements. The recursive calls will have the same characteristics, so the running time complexity will be **O(n2).**

**Problem #2:**

(a) Implement (in Java) the RadixSort algorithm to sort in increasing order an array of integer keys.

public void radixSort(int arr[])

In your implementation you must consider that each key contains only even digits (0, 2, 4, 6, and 8). Your program must detect the case of odd digits in the keys, and, in this case, abort.

Note: To storage and process the bucket lists, use an ArrayList structure.

The method is in different file outside this document.

(b) What is the running time complexity of your radixSort method? Justify.

The Total time complexity is O(n). Assuming that the number of digits in the elements of the input array and the radix number are small numbers the loops that use those constants as references can be considered O(c). The method uses loops to copy the data from the input array to the buckets and then to the output, each loop is independent O(n).