**CSCI 4041, Spring 2019, Written Assignment 3**

Due Tuesday, 2/12/19, 1:00 PM (submission link on Canvas)

Group17 member:

Thomas Tang - tang0376

Jackie Ji - ji000011

Song Liu - liux4169

This is a collaborative assignment; you may work in a group of 1-3 students. However, you may not consult or discuss the solutions with anyone other than the course instructor, the TAs, or the other members of your group, nor may you use material found from outside sources as part of your solutions. In addition, if you do choose to work in a group, each group member must participate in coming up with the solution to each problem, and must be able to explain the group’s answer if asked: dividing the problems amongst the group members is not acceptable.

Complete the following problems and submit your solutions in a single pdf file to the Written Assignment 3 submission link on Canvas. If you’re working in a group, only one person should submit your answers, but make sure that you include the name and x500 of each group member at the top of the file, and that you are all in one of the WA3 Groups in Canvas. Typed solutions are preferred, but pictures or scans of a handwritten assignment in pdf form are acceptable so long as your solutions are clearly legible.

This assignment contains 4 problems, and each is worth 10 points, for a total of 40 points.

Your solutions to these problems must be clearly explained in a step-by-step manner; for most problems, the explanation will be worth far more points than the actual answer.

1. Show the operation of Quicksort (non-randomized) on the array [5, 10, 7, 8, 9, 6, 11, 12]. You must show the following for each call made to Partition:
   * The values of p and r
   * The pivot chosen
   * The state of array A after the call to Partition completes.

For example, for the first call to Partition:

p = 1, r = 8, pivot = 12, A = [5, 10, 7, 8, 9, 6, 11, 12].

p = 1, r = 7, pivot = 11, A = [5,10,7,8,9,6,11,12]

p = 1, r = 6, pivot = 6, A = [5,10,7,8,9,6,11,12]

p = 3, r = 6, pivot = 10, A = [5,6,7,8,9,10,11,12]

p = 3, r = 5, pivot = 9, A = [5,6,7,8,9,10,11,12]

p = 3, r = 4, pivot = 8, A = [5,6,7,8,9,10,11,12]

1. Using your answer to problem 1 as a guide, find a tight (big-Θ) bound on the runtime of Insertion Sort, Merge Sort, and Quicksort for the case where the input array is already sorted in increasing order, except that two elements somewhere in the array are swapped. You must justify your answer for each algorithm.

“It’s almost the same as the sorted order case” is not a sufficient justification; if you’re going to draw a comparison, you must explain how much having two elements swapped could change the runtime from the fully sorted case, in terms of n, the number of elements in the array. You also can’t just pick which two elements are swapped: your analysis must be valid for ANY two elements.

Insertion sort: Θ(n)

Let A be the list with length of n.

Let S = smaller number that was swapped.

Let L = larger number that was swapped.

Let d = the distance between S and L.

Correct order: \*\*\*\*\*\*S\*\*\*\*\*\*\*L\*\*\*\*

Swapped : \*\*\*\*\*\*L\*\*\*\*\*\*S\*\*\*\*\*

The inner loop will not be accessed until it meets the number S. After that, the algorithm will swap S one step towards L in (d) times out loop, because the inner loop will only swap once (one step forward) each time. After it meets the number L, it will swap with S. Then the algorithm will swap L one step towards the original position of S in (d-1) times inner loop. In total, the while loop will run (2d-1) times.

d<n, so 2d-1<2n-1,and 0\*n<=2d-1<=2n-1 for n>=1

Because it is total time in all for-loop, the time complexity is that n+2n-1 = 3n-1.Thus, the big-Θ is n.

Merge Sort: Θ(n\*lg(n))

Because the time complexity does not change whether it is the worst case or not. It is always n\*lg(n).

Quicksort: Θ(n^2)

Let A be the list with length of n.

Let S = smaller number that was swapped.

Let L = larger number that was swapped.

Let d = the distance between S and L.

Correct order: \*\*\*\*\*\*S\*\*\*\*\*\*\*L\*\*\*\*

Swapped : \*\*\*\*\*\*L\*\*\*\*\*\*S\*\*\*\*\*

Case 1: L is the last one

Correct order: \*\*\*\*\*\*S\*\*\*\*\*\*\*\*\*\*\*L

Swapped : \*\*\*\*\*\*L\*\*\*\*\*\*\*\*\*\*\*S

After the first partition \*\*\*\*\*\*S\*\*\*\*\*\*\*\*\*\*L

After the first partition, the list will become sorted list, which become the worst case. Let the index of smaller number before swapped be x. Then the time complexity for the first sorted part listis (x-1)^2, and for the other one is (n-x-1)^2. Thus, the time complexity is (x-1)^2+(n-x-1)^2.

Since x <n, (x-1)^2+(n-x-1)^2 < 2n^2, and (x-1)^2+(n-x-1)^2 > 0\*n^2 for n >=1, the big-theta for this case is Θ(n^2).

Case2: L is not the last one.

The partition will run with r-- until it meets S which has the same steps as the worst case. Then it will become sorted order after the swap, and it is similar with case1. As a result, its big theta will be Θ(n^2)

1. (Adapted from Exercise 2.3-7 in the textbook) Write pseudocode for a Θ(n lg n)-time algorithm that, given an array A of n integers and another integer x, determines whether or not there exist two elements in A whose sum is exactly x. Explain informally why your algorithm is Θ(n lg n)-time. You may use calls to the Merge-Sort algorithm in the textbook as part of your pseudocode. Your algorithm should return True or False.

(Hint: Let p be the smallest element of A, r be the largest element of A, and z be any other element. If p + r < x, what does that say about p + z? If p + r > x, what does that say about z + r?)

searchSum(A,x):

Merge-Sort(A,1,A.length)

j = A.length

i = 1

while i<j:

if A[j]+A[i] == x:

return True

else if A[j]+A[i] < x:

j--

else:

i++

return False

This algorithm has time complexity of (n\*lgn +n) for the worst case, for merge sort always has big-theta (n\*lgn) and the single while loop has big-theta of n. As a result, its big-theta is Θ(n\*lg(n)).

1. (Adapted from section 6.1) For each of the following arrays, could the array represent a min heap? If no, explain why not.
   1. A = [2, 15, 2, 1, 6, 5, 20, 20]
   2. B = [4, 5, 7, 18, 5, 15]
   3. C = [1, 8, 14, 19, 15, 12, 15]
   4. D = [42, 42, 42, 42, 42, 42, 42, 42, 42, 42]

AC cannot represent in a min heap.

A: 2 C:1

15 2 8 14

1 6 5 20 19 15 12 15

20

As you can see, the 1 and 6 is smaller than 15 in A, and 12 is smaller than 14 in C.Both of them does not follow the property of min-heap.