CSC175 Practice Assignment 6 Spring 2019 Name Solutions

1) Use the Euclidean Algorithm to find: (show each step of the algorithm)

a) GCD(540, 395)

540 = 1(395) + 145
395 = 2(145) + 105
145 = 1(105) + 40
105 = 2(40) + 25
40 = 1(25) + 15
25 = 1(15) + 10
15 = 1(10) + 5
10 = 2(5) +0

Therefore, the GCD(540, 395) = 5

b) GCD(675, 56)

675 = 12(56) + 3
56 = 18(3) + 2
3 = 1(2) +1
2 = 2(1) + 0

Therefore, the GCD(675, 56) = 1

2) Consider the sorted list 9 16 17 29 38 54 78

a) What is the most steps it could take to find a number if we traveled through the list one at a time? Why?
7 because we could be looking for 78 (the last number) and we could have started looking at the list from the beginning.

b) What is the most steps it could take to find a number of we traveled through the list using binary search? Why?
3 because if we split the list of 7, then the list of 3, then we only have one number left.

c) Use binary search to find 38. Explain each step of the algorithm.
We first check the middle number which is 29. Since 38>29, we consider the right side of the list: 38 54 78.
We look at 54 (the new middle of the list). Since 38<54, we look at the left side of the list which is 38. We look at 38 which is our number. It took three iterations as determined in part b.
Note: $\log_2 7=(\log 7)/(\log 2)=2.8$ which shows we need up to 3 iterations to guarantee that we could find our number in a sorted list.

3) Give the Big O runtime for each algorithm. Explain your answer.

a) Begin Check_Algorithm (List of numbers A)

```
Enter N (length of A)
For i = 1 to N
For k = i to N
If A[i]<A[k]
   print A[k]
   End If
End For
End For
```

End Check_Algorithm

Since there are two nested FOR loops, we have $O(N^2)$. Note: We actually have

$$N + N - 1 + N - 2 + \cdots + 1 = \frac{N(N+1)}{2} = \frac{N^2 + N}{2}$$

steps as seen in one of the videos which is asymptotic to N^2 (quadratic).

b) Begin Check_Again_Algorithm (List of numbers A)

```
Enter N (length of A)
For i = 2 to N
If A[i]<A[1]
   print A[i]
   End If
End For
For k = 3 to N
    If  A[k]<A[2]
    print A[k]
    End If
End For
```

End Check_Again_Algorithm

We have two FOR loops but they are not nested. We have N-1 steps in the first FOR loop and then N-2 in the second FOR loop. Thus, the run time is N-1+N-2 = 2N-3 (linear) so we say it is O(N).

c) Begin Summer_Algorithm (List of numbers A where the first entry (input) is the month (output): 1=Jan, 2 = Feb, etc.)

```
Enter N (length of A)
If A[1]=6
   Print "It is summer!"
Else If A[1]=7
   Print "It is summer!"
Else If A[1]=8
   Print "It is summer!"
Else
   Print "It is not summer! ☹"
End If
```

End Summer_Algorithm

We have at most 3 comparisons regardless of the length of A so this is O[1].

4) Explain what is happening for each of the following algorithms. Be as thorough as possible.

a) The Check_Algorithm in 3a above

We are comparing each element with all of the other elements that follow it in the list. We start with the first element A[1]. We compare it to itself which is clearly not greater than it. Then we proceed by comparing A[1] with all elements of the list A[2] through A[N]. If any of the values are greater than A[1], the value is printed.

Then we look at A[2] and check if itself or any of the elements following it are larger and print them if they are. We do the same with each element of the list, printing all elements that are larger than the proceeding values. Each time a value is printed, we know how many numbers before it in the list are less than it.

(You would not need an example…this is just to help you understand the algorithm: Example List: 9, 4, 6, 10, 2 Since N=5, then i=1 to 5 and k = i to 5. A[1] = 9 so we compare it to the rest of the list. Only 10 is greater so the only number printed is 10. Then A[2] = 4. Since 6 and 10 are larger, then both would be printed. Remember, we are only checking values starting with the current position to the right so we don't compare 4 to the 9 at the front of the list. Next, A[3] = 6. The only number bigger than it to the right is 10 so it is printed. Next, A[4]=10. Nothing bigger than it to the right. Finally, A[5] = 2. Since it is the last element in the list, we wouldn't have any number greater than it. The final print out would be: 10 6 10 10 Since 10 is printed three times, that means it is larger than three elements that preceded it. The element 6 only has one element preceding it that it is larger than. One other note, this is not an efficient algorithm since we are checking a term with itself which is a waste of time. However, the runtime is easier to determine for this setup.)

b) The Check_Again_Algorithm in 3b above

For the first FOR loop, we are checking the 2nd through the nth term with the first term. If any of the terms are less than the first term, it is printed. For the second FOR loop, we are checking the 3rd through the nth term with the second term. If any of the terms are less than the second term, it is printed.

(Again, you would not need an example: Example List: 9, 4, 6, 10, 2
Since N=5, then i=2 to 5. For the first FOR loop, A[1] = 9 so we compare it to the rest of the list. Since 4, 6, and 2 are all less than 9, they would be printed. For the second FOR loop, A[2] = 4 so we compare it to the rest of the list. Since 2 is the only number less than 4, we print 2. The final print out would be: 4 6 2 2 Note, depending how you would code it, the numbers could appear in a row with spaces, in a row without spaces, or in a column.)

5) Consider the list 12 4 19 21 8 5

a) Explain how bubble sort works in your own words. Use bubble sort to sort the list. Show each swap.

We will compare the first element with the second element and swap if the first is greater than the second. Then we will compare the second with the third, third with the fourth, and so on, swapping if the former is greater than the latter. Once we get to the end, we start over and do the same process until no swap is made as we traverse through the list, meaning the list is sorted. Note, if programming this algorithm, each time we go through the list, the largest remaining number in the unsorted list bubbles to the top.

12 4 23 21 8 5
4 12 **23 21** 8 5
4 12 21 **23 8** 5
4 12 21 8 **23 5**
4 12 **21 8** 5 23
4 12 8 **21 5** 23
4 **12 8** 5 21 23
4 8 **12 5** 21 23
4 **8 5** 12 21 23
4 5 8 12 21 23

b) Explain how selection sort works in your own words. Use selection sort to sort the list. Show each swap.

We will look for the smallest number in the list and swap it with the number in the first element in the list. Then we start in the second position (since the first position has the smallest number), find that smallest remaining number, and swap it with the second number in the list. We continue this process until we have a sorted list.

12 **4** 23 21 8 5
4 | **12** 23 21 8 **5**
4 5 | **23** 21 **8** 12
4 5 8 | **21** 23 **12**
4 5 8 12 | **23** **21**
4 5 8 12 21 23

6) Answer each question.

a) What is the best case runtime for bubble sort? When does that occur?

Ω(N) when the list is already sorted.

b) What is the worst case runtime for bubble sort? When does that occur?

$O(N^2)$ when the list is not sorted.

c) Why is selection sort $O(N^2)$ for both best and worst case?

Even if the list is already sorted, we need to go back through the list each time to find the new smallest to swap to the front of the unsorted portion of the list.