Toshanraju Vysyaraju RUID: 184005155

Written Assignment 5

Problem 1

1. Selection Sort

   `[C, Q, S, A, X, B, T]` - Original (Unsorted)

   6 Comparisons

   `[A, Q, S, C, X, B, T]`

   5 Comparisons

   `[A, B, S, C, X, Q, T]`

   4 Comparisons

   `[A, B, C, S, X, Q, T]`

   3 Comparisons

   `[A, B, C, Q, X, S, T]`

   2 Comparisons

   `[A, B, C, Q, S, X, T]`

   1 Comparison

   `[A, B, C, Q, S, T, X]` - Final (Sorted)

2. Insertion Sort

   `[C, Q, S, A, X, B, T]` - Original (Unsorted)

   1 Comparison

   `[C, Q, S, A, X, B, T]`

   1 Comparison

   `[C, Q, S, A, X, B, T]`

   3 Comparisons

   `[A, C, Q, S, X, B, T]`

   1 Comparison

   `[A, C, Q, S, X, B, T]`

   5 Comparisons

   `[A, B, C, Q, S, X, T]`

   2 Comparisons

   `[A, B, C, Q, S, T, X]` - Final (Sorted)

Problem 2

1. Determine if two arrays have no elements in common.
    a. Algorithm

```
READ first array and SET TO array1
READ second array and SET TO array2
FOR each value in array1
      FOR each value in array2
            IF value in array1 EQUALS value in array2
                  RETURN true
            ENDIF
      ENDFOR
ENDFOR
RETURN false
```

    b. Factors

    n - size of array1

    m - size of array2

    c. Number of Operations per Step

```
READ first array and SET TO array1 - 1 operation
READ second array and SET TO array2 - 1 operation
FOR each value in array1 - n operations
      FOR each value in array2 - m operations
            IF value in array1 EQUALS value in array2 - 1 Operation
                  RETURN true - 1 Operation
            ENDIF
      ENDFOR
ENDFOR
RETURN false - 1 Operation
```

    d. Total Operations

    $f(n, m) = 2 \cdot n \cdot m + 3$

    e. Best Case

    This occurs when the first element of both arrays are the same. In this case, there will be 6 operations done. The Big O is O(1).

    f. Worst Case

    This occurs when there are no common elements in the two arrays. In this case, there will be $2 \cdot n \cdot m + 3$ operations. The Big O is $O(n \cdot m)$.

2. Counting the total number of characters that have a duplicate within a string.
    a. Algorithm

```
READ input and SET TO input
SET wordArray AS array of characters of input
SET count to 0
FOR character in wordArray
      IF character DOES NOT EQUAL “ ”
            SET currentCount TO 0
            FOR char in wordArray
                  IF char EQUALS character
                        ADD 1 to currentCount
                        SET character IN wordArray TO “ ”
                  ENDIF
            ENDFOR
      ENDIF
      IF currentCount IS GREATER THAN 1
            ADD currentCount to count
      ENDIF
ENDFOR
RETURN count
```

    b. Factors

    n - the length of the entered string

    c. Number of Operations per Step

```
READ input and SET TO word - 1 operation
SET wordArray AS array of characters of input - 1 operation
SET count to 0 - 1 operation
FOR character in wordArray - n operations
      IF character DOES NOT EQUAL “ ” - 1 operation
            SET currentCount TO 0 - 1 operation
            FOR char in wordArray - n operations
                  IF char EQUALS character - 1 operation
                        ADD 1 to currentCount - 1 operation
                        SET character IN wordArray TO “ ” - 1 operation
                  ENDIF
            ENDFOR
      ENDIF
      IF currentCount IS GREATER THAN 1 - 1 operation
            ADD currentCount to count - 1 operation
      ENDIF
ENDFOR
RETURN count - 1 operation
```

    d. Total Operations

    Best Case: $f(n) = 4$

    Worst Case: $f(n) = 1 + 1 + 1 + n + 1 + 1 + n \bullet n + 1 + 1 + 1 = n \bullet n + n + 7$

    e. Best Case

    The best case occurs when there is an empty string. In this case the Big O would be O(1).

    f. Worst Case

    The worst case occurs when there are no duplicates in a long string. In this case the Big O would be $O(n^2)$.

3. Finding a row where every entry is ‘x’ in a 2-D array.
    a. Algorithm

```
READ input and SET TO array
FOR each row in array
SET present EQUAL TO true
      FOR each column in row in array
            IF item IS NOT EQUAL TO x
                  SET present EQUAL TO false
            ENDIF
      ENDFOR
      IF present IS EQUAL TO true
            DISPLAY row
            RETURN
      ENDIF
ENDFOR
```

    b. Factors

    r - number of rows

    c - number of columns

    c. Number of Operations per Step

```
READ input and SET TO array - 1 operation
FOR each row in array - r operations
SET present EQUAL TO true - 1 operation
      FOR each column in row in array - c operations
            IF item IS NOT EQUAL TO x - 1 operation
                  SET present EQUAL TO false - 1 operation
            ENDIF
      ENDFOR
      IF present IS EQUAL TO true - 1 operation
            DISPLAY row - 1 operation
            RETURN - 1 operation
      ENDIF
ENDFOR
```

    d. Total Operations

    Best Case: $f(r, c) = 1 + 1 + 1 + c + c + 1 + 1 + 1 = 2 \cdot c + 6$

    Worst Case: $f(r, c) = 1 + r + r \cdot c + r \cdot c + 1 + r + 1 + 1 = 2 \cdot r \cdot c + 2 \cdot r + 4$

    e. Best Case

    The best case is when the first row in the 2D array is all x’s. The Big O for this would be O(c).

    f. Worst Case

    The worst case is when the last row in the 2D array is all x’s. The Big O for this would be O(r•c).