ENSF 594 – Assignment 3

By Graydon Hall (30142310)

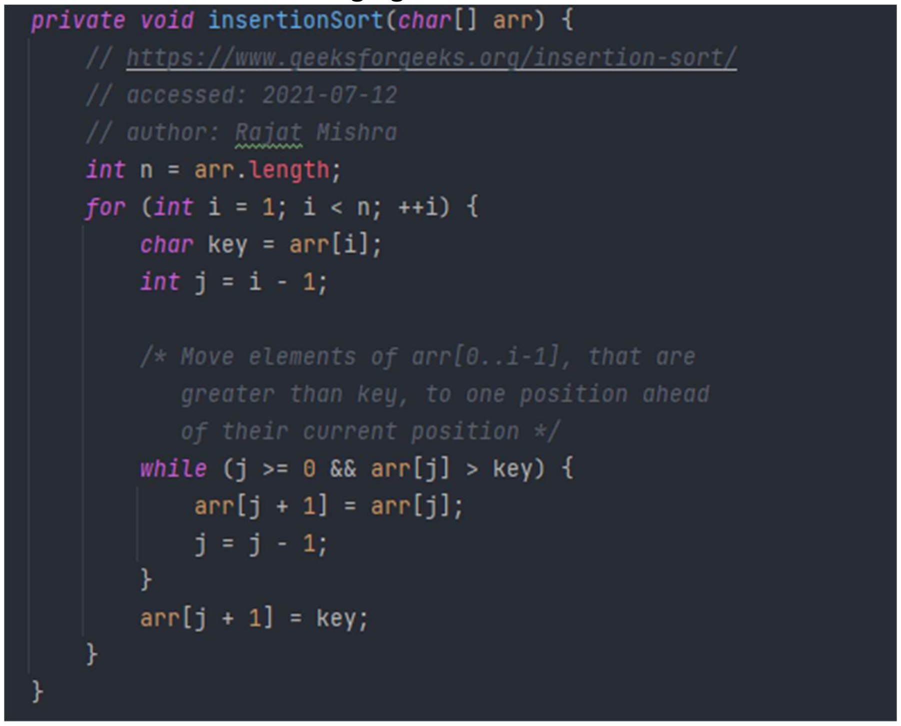
Complexity analysis for my program

Question:

What is the worst-case complexity of your algorithm when checking if two words are anagrams of each other? Express this using big-O notation and use the variable k to represent the number of letters in each word.

Answer:

When my algorithm compares two words to check if they are anagrams, both words are broken into an array of chars. This array of chars is then sorted using an insertion sort algorithm, which was taken from the previous assignment. The sorted array of chars is then converted back into a string, and if the two words are anagrams, the strings will be equal. The insertion sort implementation is shown in the following figure:



```
private void insertionSort(char[] arr) {
    // https://www.geeksforgeeks.org/insertion-sort/
    // accessed: 2021-07-12
    // author: Rajat Mishra
    int n = arr.length;
    for (int i = 1; i < n; ++i) {
        char key = arr[i];
        int j = i - 1;

        /* Move elements of arr[0..i-1], that are
           greater than key, to one position ahead
           of their current position */
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
```

Since the insertion sort implementation uses nested loops, it has a worst case complexity of O(n^2), for both comparisons and data moves. Therefore, the worst-case complexity of my program when checking if two words are anagrams is O(n^2). If you compare two words with k letters each, the total complexity would be $2*k^2$ in the worst case scenario.