## ENSF 338 L03 Exercise #4 Group 7

**Question 1:**
Input array size = n
Worst case scenario it takes n times to partition the array
Each time we partition, a recursive function is called.
Therefore,
[n + (n-1) + (n-2) + (n-3) + … + 2 ] = [$\frac{n(n+1)}{2}$ -1 ] = [$\frac{n^2+n+1}{2}$ -1 ] = O(n^2)

**Question 2:**
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]

Pivot = 1
partitioning:
left subarray: [1]
right subarray: [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 2
partitioning:
left subarray: [2]
right subarray: [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 3
partitioning:
left subarray: [3]
right subarray: [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 4
partitioning:
left subarray: [4]
right subarray: [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 5
partitioning:
left subarray: [5]
right subarray: [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 6
partitioning:
left subarray: [6]
right subarray: [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 7
partitioning:
left subarray: [7]
right subarray: [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 8
partitioning:
left subarray: [8]
right subarray: [8, 9, 10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 9
partitioning:
left subarray: [9]
right subarray: [9, 10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 10
partitioning:
left subarray: [10]
right subarray: [10, 11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 11
partitioning:
left subarray: [11]
right subarray: [11, 12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 12
partitioning:
left subarray: [12]
right subarray: [12, 13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 13
partitioning:
left subarray: [13]
right subarray: [13, 14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

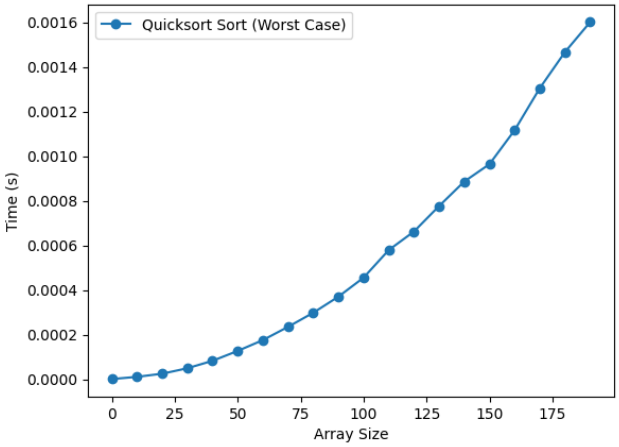
Pivot = 14
partitioning:
left subarray: [14]
right subarray: [14, 15, 16]
quick sort applied to both arrays
left is sorted already… now onto the right

Pivot = 15
partitioning:
left subarray: [15]
right subarray: [15, 16]
quick sort applied to both arrays
left is sorted already… and right is added to left as it has one element left therefore the quicksort algorithm is completed and we are left with [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16].

**Question 3:**

```
import timeit
import random
import sys
import numpy as np
import matplotlib.pyplot as plt
def quicksort(arr, low, high):
    if low < high:
        pi = partition(arr, low, high)
        quicksort(arr, low, pi - 1)
        quicksort(arr, pi + 1, high)

#partition method aided via ChatGPT
def partition(arr, low, high):
    pivot = arr[high]
    i = low - 1
    for j in range(low, high):
        if arr[j] <= pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i + 1], arr[high] = arr[high], arr[i + 1]
    return i + 1

#ordered arrays with elements valued from 1- 100 with size 5 incremented by 10 in
size each all in ascending order (sorted)
arr_sorted = [[i for i in range(i*10) ] for i in range(20)]
print(len(arr_sorted[19]))
quicksort_times = []
for x in range(20):
    #worse case quicksort: array is sorted (n^2)
    quicksort_times.append(timeit.timeit(lambda:
quicksort(arr_sorted[x],0,len(arr_sorted[x]) -1),number=1))
    print("quicksort: pass",x+1)

sizes = [(i*10)for i in range(20)]

#Insertion vs. Binary Insertion Plot
plt.plot(sizes, quicksort_times, label='Quicksort Sort (Worst Case)', marker='o')
plt.xlabel('Array Size')
plt.ylabel('Time (s)')
plt.legend()
plt.show()
```

**Question 4:**




Yes they do O(n^2)