Question 1.

a. $f(n) = n^{0.25}; g(n) = n^{0.5}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{n^{0.25}}{n^{0.5}} = \frac{1}{n^{0.25}} = 0; therefore\ g(n) grows\ faster\ than\ f(n)$$

$$f(n) = O(g(n))$$

b. $f(n) = logn; g(n) = \ln n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{\log n}{\ln n} = \frac{\log n}{\frac{\log n}{\log e}} = \log e\ ; therefore\ the\ functions\ grow\ at\ the\ same\ rate$$

$$f(n) = \theta(g(n))$$

c. $f(n) = n\log n\ ; g(n) = n\sqrt{n}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{n\log n}{n\sqrt{n}} = \frac{\log n}{\sqrt{n}}$$

$$Apply\ L'hopital\ rule$$

$$\frac{\frac{1}{n\ln 10}}{\frac{1}{2n^{\frac{1}{2}}}} = \frac{1}{n\ln 10} * \frac{2n^{\frac{1}{2}}}{1} = \frac{2}{\ln 10} * n^{-\frac{1}{2}} = \lim_{n\to\infty}\left(\frac{2}{\ln 10} * \frac{1}{\sqrt{n}}\right) = \frac{1}{\sqrt{n}} = 0;$$

$$therefore\ g(n)\ grows\ faster\ than\ f(n).$$

$$f(n) = O(g(n))$$

d. $f(n) = 2^n; g(n) = 3^n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{2^n}{3^n} = (\frac{2}{3})^n = 0$$

$$therefore\ the\ g(n) grows\ faster\ than\ f(n).$$

$$f(n) = O(g(n))$$

e. $f(n) = 2^n; g(n) = 2^{n-1}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{2^n}{2^{n-1}} = \frac{2^n}{2^{-1}2^n} = 2$$

$$therefore\ the\ functions\ grow\ at\ the\ same\ rate.$$

$$f(n) = \theta(g(n))$$

f. $f(n) = 4^n; g(n) = n!$

We assume that $f(n) = O\big(g(n)\big), we\ must\ show\ that$:

$$\exists c, n_o\ such\ that\ 0 \leq f(n) \leq cg(n)\ for\ c > 0\ and\ all\ n > n_o$$

$$Informally, we\ must\ show\ that\ g(n)\ grows\ faster\ than\ f(n)\ for\ some\ n > n_o$$

We can prove with Induction:

$Base\ Case$:

$Let\ n = 10, then$:

$f(n) = 4^{10} = 1{,}048{,}576$

$g(n) = 10! = 3{,}628{,}800$

$therefore\ 4^{10} \leq 10!, the\ base\ case\ is\ true$

$Inductive\ Hypothesis$:

$Assume\ that\ if\ 4^k \leq k!\ is\ true, then\ 4^{k+1} \leq (k+1)!, for\ k \geq 10$

$Inductive\ Step$: $we\ must\ show\ that\ 4^{k+1} \leq (k+1)!$

$(k+1)! = (k+1) * k! \quad (definition\ of\ factorial)$

$k! \geq 4^k \ \ (induction\ hypothesis)$

$We\ can\ also\ observe\ that\ the\ quantity\ (k+1)\ is\ \geq 4, since\ k \geq 10, therefore$:

$(k+1) * k! \geq 4^k * 4$

$(k+1) * k! \geq 4^{k+1} \quad (exponent\ product\ rule), therefore$

$(k+1)! \geq 4^{k+1}\ is\ true, which\ is\ what\ we\ were\ trying\ to\ prove$

$Therefore\ when\ \ c = 1\ and\ n_o \geq 10, f(n) = O\big(g(n)\big).\ \ QED$

Question 2. Prove or disprove the following:

(a) If $f_1(n) = \Omega(g(n))$ and $f_2(n) = O(g(n))$ then $f_1(n) = \Theta(f_2(n))$.

$$Let\ f_1(n) = n^2; g(n) = n;\ f_2(n) = 1$$

$$Let\ the\ constants\ c_1 = c_2 = 1\ and\ n_1 = n_2 = 2, we\ substitue\ these\ in\ the\ following:$$

$$0 \le c_1 g(n) \le f_1(n), then\ 0 \le 1 * n \le n^2\ for\ all\ n \ge 2, therefore\ f_1(n) = \Omega(g(n))$$

$$0 \le f_2(n) \le c_2 g(n), then\ 0 \le 1 \le 1 * n\ for\ all\ n \ge 2, therefore\ f_2(n) = O(g(n))$$

$$If\ we\ assume\ that\ f_1(n) = \theta(f_2(n)), then\ by\ definition, f_1(n) = O(f_2(n))\ and\ f_1(n) = \Omega(f_2(n))$$

$$However, performing\ asympotic\ analysis:$$

$$\lim_{n\to\infty} \frac{f_1(n)}{f_2(n)} = \frac{n^2}{1} = \infty, f_1(n)\ grows\ faster\ than\ f_2(n), then$$

$$f_1(n) = O(f_2(n))\ is\ false, therefore\ \ f_1(n) = \theta(f_2(n))\ is\ false.\ \ QED.$$

(b) $If\ \ f_1(n) = O(g_1(n))\ and\ f_2(n) = O(g_2(n))then\ f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$

We need to prove that:

$$\exists c, n\ such\ that\ 0 \le f_1(n) + f_2(n) \le c g_1(n) + c g_2(n)\ for\ c > 0\ and\ n > n_o$$

By definition:

$f_1(n_1) = O(g_1(n_1)) =>$

$$\exists\ c_1, n_1\ such\ that\ 0 \le f_1(n_1) \le c_1 g_1(n_1)\ for\ c_1 > 0\ and\ n_1 > n_o$$

$$and$$

$f_2(n_2) = O(g_2(n_2)) =>$

$$\exists\ c_2, n_2\ such\ that\ 0 \le f_2(n_2) \le c_2 g_2(n_2)\ for\ c_2 > 0\ and\ n_2 > n_o$$

Therefore:

$$f_1(n_1) + f_2(n_2) \le c_1 g_1(n_1) + c_2 g_2(n_2)$$
$$Suppose\ we\ let, c_o = c_1 + c_2\ and\ n_o = \max(n_1, n_2)$$
$$then\ c_1 g_1(n) + c_2 g_2(n) \le c_o(g_1(n) + g_2(n))\ for\ n \ge n_o$$
$$Therefore\ there\ exists\ a\ c_o\ and\ n_o\ such\ that:$$
$$f_1(n) + f_2(n) = O(g_1(n) + g_2(n))\ when\ n \ge n_o$$

4. Insert Sort and Merge Sort running time analysis.

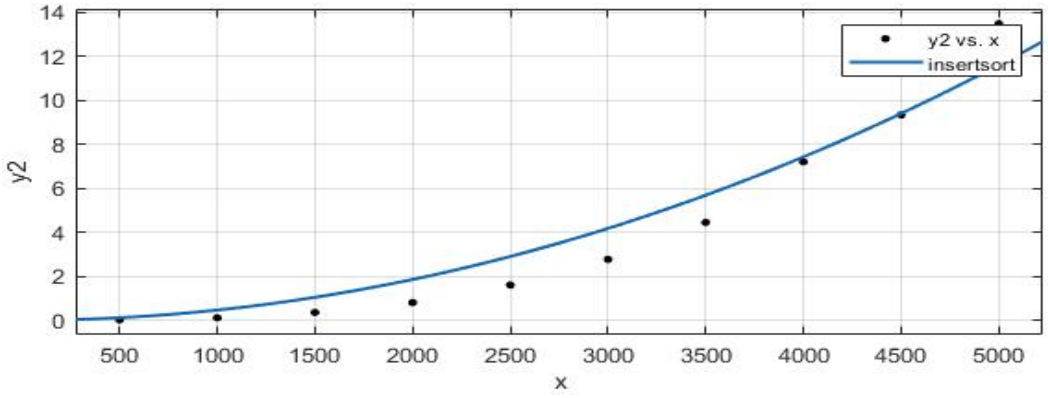
B). Running times

Insertion Sort Algorithm	
Array Size (n)	Running Time(s)
500	0.015104
1000	0.11206
1500	0.352841
2000	0.803589
2500	1.598262
3000	2.771835
3500	4.445526
4000	7.195229
4500	9.333934
5000	13.48005

Merge Sort Algorithm	
Array Size (n)	Running Time(s)
5000	0.002678
10000	0.005489
15000	0.008693
20000	0.011938
25000	0.015271
30000	0.018694
35000	0.021815
40000	0.02566
45000	0.028218
50000	0.031701

C.) Plot data and fit curve

Insert Sort Curve:

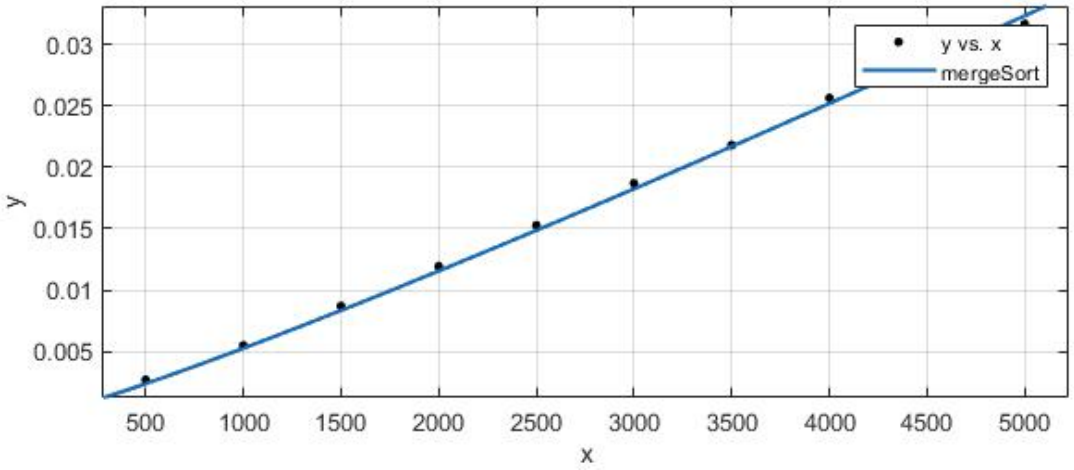


Equation of the curve:

$$f(x) = a * x^2$$

$$More\ specifically: f(x) = 4.64e^{-07} * x^2$$

Merge Sort Curve:

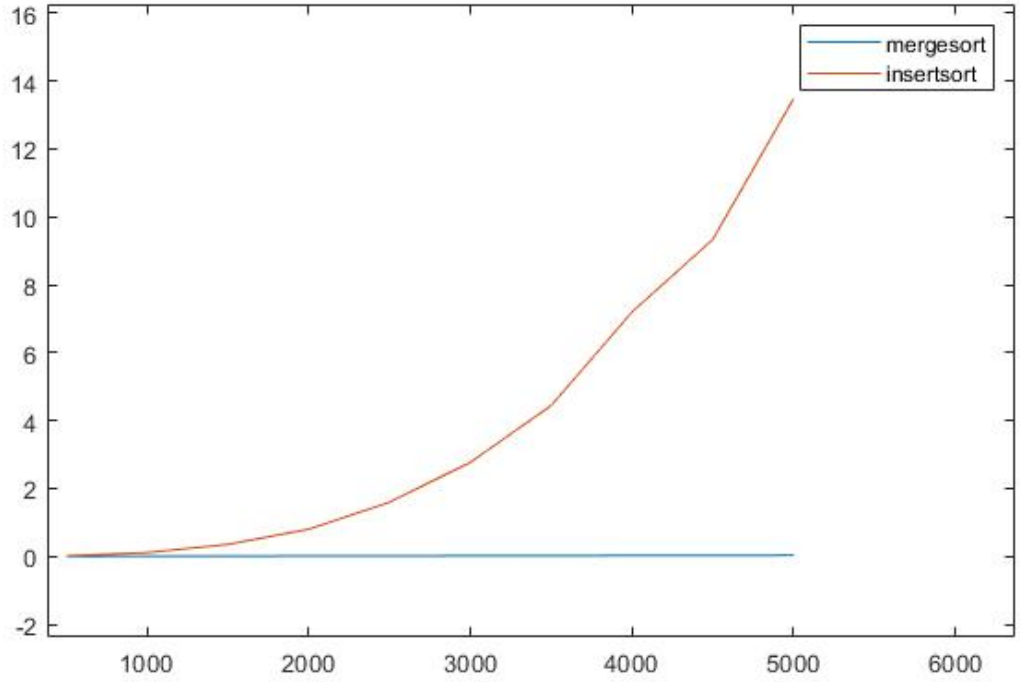


Equation of the Curve:

$$f(x) = x * \log(x)$$

$$More\ specifically: f(x) = 7.559e^{-07} * \log(x)$$

D.) Combined Plot:



E.) Comparison

The merge sort algorithm is clearly much faster insertion sort. In the largest array in the trial, 50000, the running time of merge sort was 0.37s vs 1431s of insertion sort. The merge sort algorithm completed in less than one second for all time trials. For this scenario, the divide and conquer strategy of the merge sort algorithm is clearly superior to the insertion sort algorithm.

insertTime Code:

```
#!/usr/bin/env python
import time
from random import *

'''
   Author: Francis C. Dailig
   Date: 1/13/2019
   File: insertTime
   Description: This is the python script for the insertion sort algorithm
'''


#################################################################################################
#Function Name:  getArray()                                                                     #
#Description:  The getArray() function will take as an argument, an array and a filename       #
#It will copy the numbers from the filename into the array.                                     #
#################################################################################################
def getArray(array, fileName):

   with open(fileName) as data_File:
      for line in data_File:
         for i in line.split():
            array.append(int(i))


#################################################################################################
#Function Name:  insertionSort()                                                                #
#Description:  The insertionSort() will sort an array.  It takes one argument - array          #
#################################################################################################
def insertionSort(arr, size):
   for i in range(size-1):
      key = i + 1
      value = arr[key]

      while key > 0 and value < arr[key - 1]:
         arr[key] = arr[key - 1]
         key -= 1

      arr[key] = value


#################################################################################################
#Function Name:  runTimeTrial()                                                                 #
#Description:  This funciton will run time trials on the arrays of increasing size.            #
#################################################################################################
def runTimeTrial(lengthArray,  timeArray):
   testArray = []
   for i in range(len(lengthArray)):
      print("running trial: %i" % i)
      print("The size of the array is: %i" % lengthArray[i])
      for i in range(lengthArray[i]):
         testArray.append(randint(0,10000))

      start = time.time()
      insertionSort(testArray, len(testArray))
      end = time.time()
      timeArray.append(end-start)
      print("time to complete: %f" % (end-start))

#################################################################################################
#Function Name:  main()                                                                         #
#Description:  This is the main() function.                                                     #
#################################################################################################
```

```
def main():
    print("Running time trials for the insertion sort algorithm....")

    dataArray = [500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000]
    timeArray = []

    runTimeTrial(dataArray, timeArray)



if __name__ == "__main__": main()
```

mergeTime Code:

```
#!/usr/bin/env python
import time
from random import *

'''
   Author: Francis C. Dailig
   Date: 1/13/2019
   File: mergeTime
   Description: Python Script for the merge sort algorithm
'''

number_Array = []
length = 0

#########################################################################################################
#Function Name:  getArray()                                                             #
#Description:  The getArray() function will take as an argument, an array and a filename       #
#It will copy the numbers from the filename into the array.                              #
#########################################################################################################
def getArray(array, fileName):
   with open(fileName) as data_File:
      for line in data_File:
         for i in line.split():
            array.append(int(i))

   print(array)

#########################################################################################################
#Function Name:  merge()                                                                #
#Description:  The merge() function is a subroutine called by the mergesort() function to      #
#combine two array.  It takes three argument - an array, left subarray, and right subarray.     #
#########################################################################################################
def merge(array, leftArray, rightArray):
   leftSize = len(leftArray)
   rightSize = len(rightArray)

   i,j,k = 0,0,0

   while j < leftSize and k < rightSize:
      if leftArray[j] < rightArray[k]:
         array[i] = leftArray[j]
         j += 1
      else:
         array[i] = rightArray[k]
         k += 1
      i += 1

   while j < leftSize:
      array[i] = leftArray[j]
      i += 1
      j += 1

   while k < rightSize:
      array[i] = rightArray[k]
      i += 1
      k += 1

#########################################################################################################
#Function Name:  mergeSort()                                                            #
#Description:  The mergeSort() will sort an array.  It takes one argument - array.           #
#########################################################################################################
def mergeSort(array):
   length = len(array)
```

```
    if length == 1:
        return
    else:
        mid = length/2
        leftSize = mid
        rightSize = mid + length % 2

        left = [0] * leftSize
        right = [0] * rightSize

        for i in range(leftSize):
            left[i] = array[i]

        for i in range(rightSize):
            right[i] = array[mid+i]

        mergeSort(left)
        mergeSort(right)
        merge(array, left, right)


#############################################################################################################
#Function Name:  runTimeTrial()                                                                             #
#Description:  This function will run time trials on arrays of increasing size.                             #
#############################################################################################################
def runTimeTrial(lengthArray, timeArray):
    testArray = []
    for i in range(len(lengthArray)):
        print("running trial: %i" % i)
        print("The size of the array is: %i" % lengthArray[i])
        for i in range(lengthArray[i]):
            testArray.append(randint(0,10000))

        start = time.time()
        mergeSort(testArray)
        end = time.time()
        timeArray.append(end-start)
        print("time to complete: %f" % (end-start))
        testArray = []


#############################################################################################################
#Function Name:  main()                                                                                     #
#Description:  This is the main() function.                                                                 #
#############################################################################################################
def main():
    print("Running time trials for the mergeSort algorithm...")
    numberArray = []

    dataArray = [500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000]
    timeArray = []
    runTimeTrial(dataArray, timeArray)

    print(timeArray)

if __name__ == "__main__": main()
```