CS325 - HW1

1) Using limit method

-
    a. f(n) is O(g(n))
    b. f(n) is Θ(g(n))
    c. f(n) is O(g(n))
    d. f(n) is O(g(n))
    e. f(n) is O(g(n))
    f. f(n) is Ω(g(n))
    g. f(n) is Θ(g(n))
    h. f(n) is Ω(g(n))

2)

a. 0 + 1 + 2 + 3 + 4 + … + n-1

   {n(n+1)}/2 => sum notation
   Then

   {(n-1)n}/2
   Then O(n^2) notation

b. I = 2, 2^2, 2^4, 2^8 ...
   Total = 1, 2, 3, 4 ...

   So, O(log(log(n)))

c. 1st loop => ½ * n
   2nd loop => m
   3rd loop => n*m

   Then add all loop statement = O(½ n + m + mn) then = O(mn)

3) see the attached files

4)

a) See the attached files

b) insertTime.py

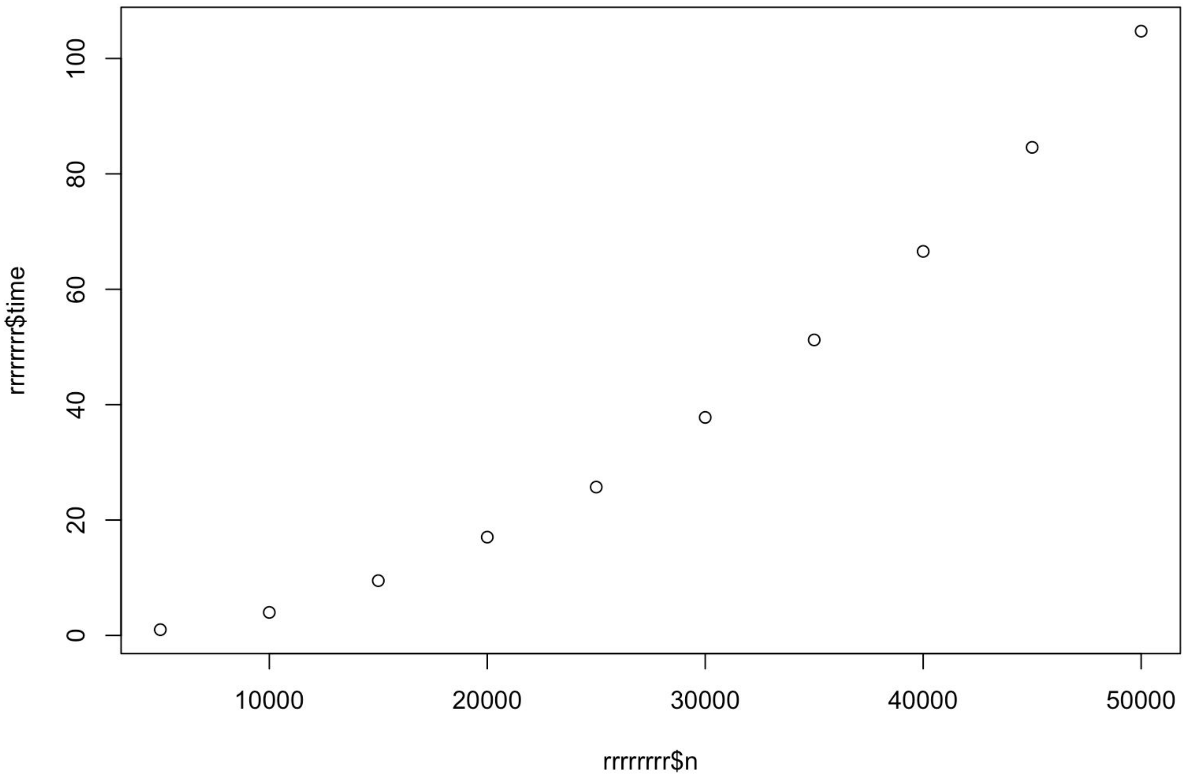
n	time
n: 5000	time: 0.9941917260487875
n: 10000	time: 3.9892350832621255
n: 15000	time: 9.489851474761963
n: 20000	time: 17.035858154296875
n: 25000	time: 25.706602255503338
n: 30000	time: 37.78681135177612
n: 35000	time: 51.21909252802531
n: 40000	time: 66.56304383277893
n: 45000	time: 84.59582694371541
n: 50000	time: 104.73469376564026

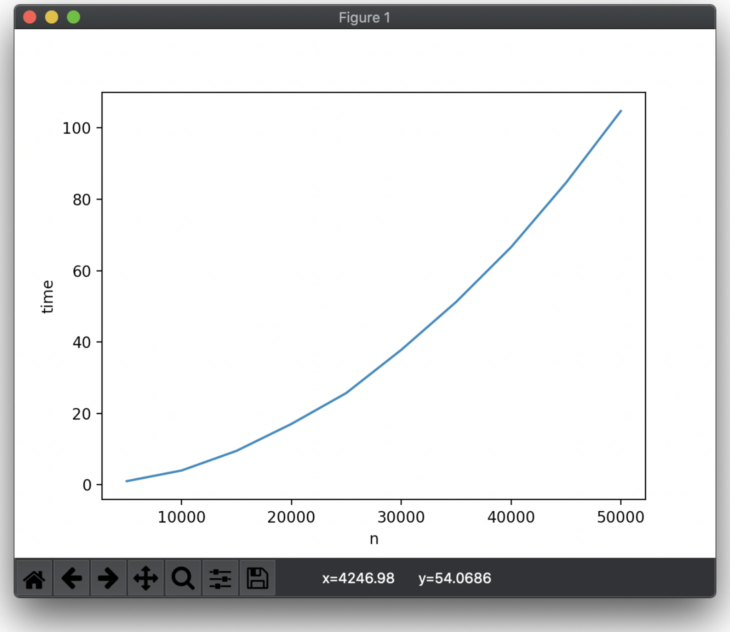
MergeTime.py

n	time
n: 5000	time: 0.02212969462076823
n: 10000	time: 0.04770596822102865
n: 15000	time: 0.07779479026794434
n: 20000	time: 0.10632705688476562
n: 25000	time: 0.133196751276652
n: 30000	time: 0.15685105323791504
n: 35000	time: 0.1953125
n: 40000	time: 0.22485836346944174
n: 45000	time: 0.2595655918121338

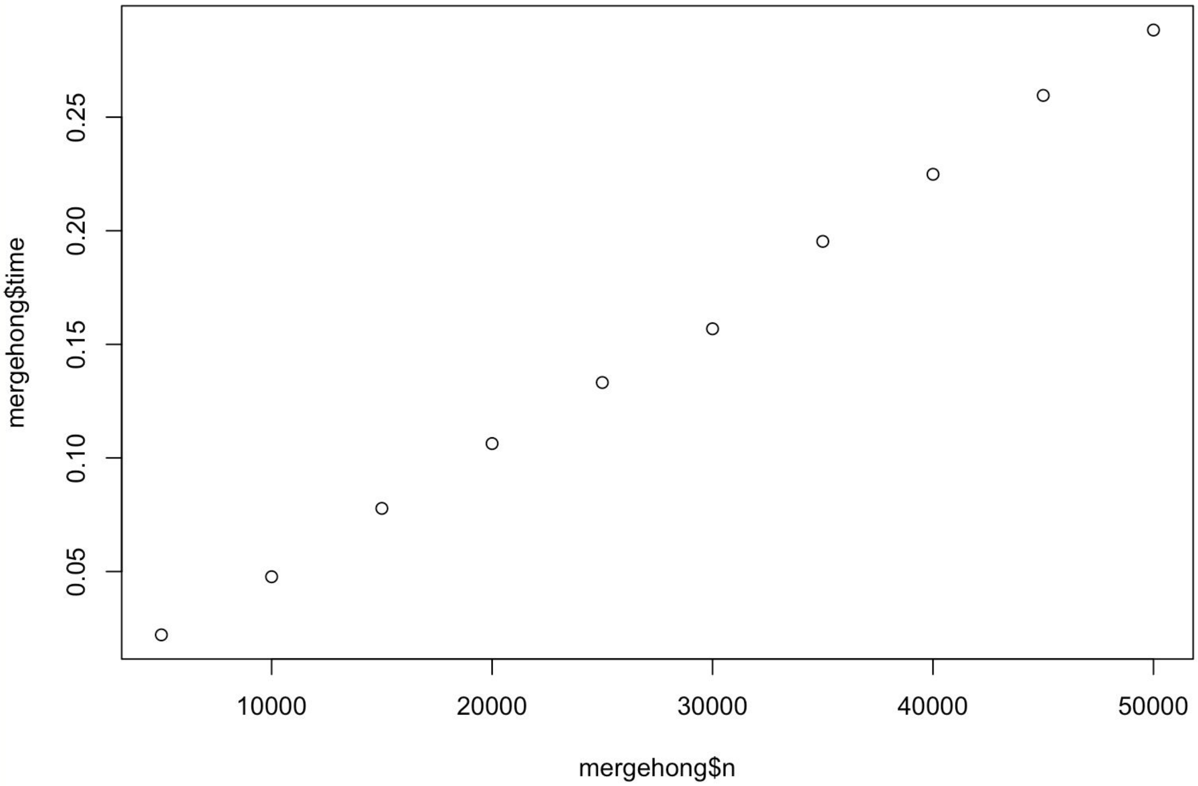
n: 50000	time: 0.2883426348368327

c) INSERTION SORT PLOT





MERGE SORT PLOT



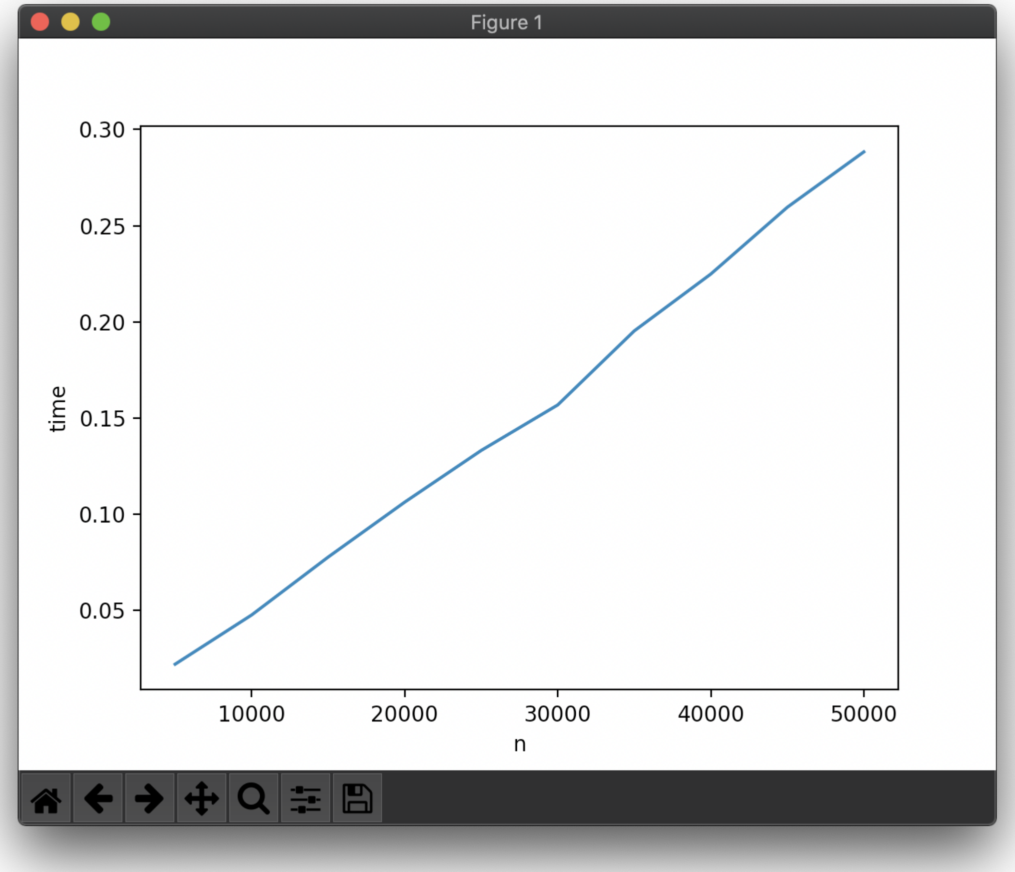
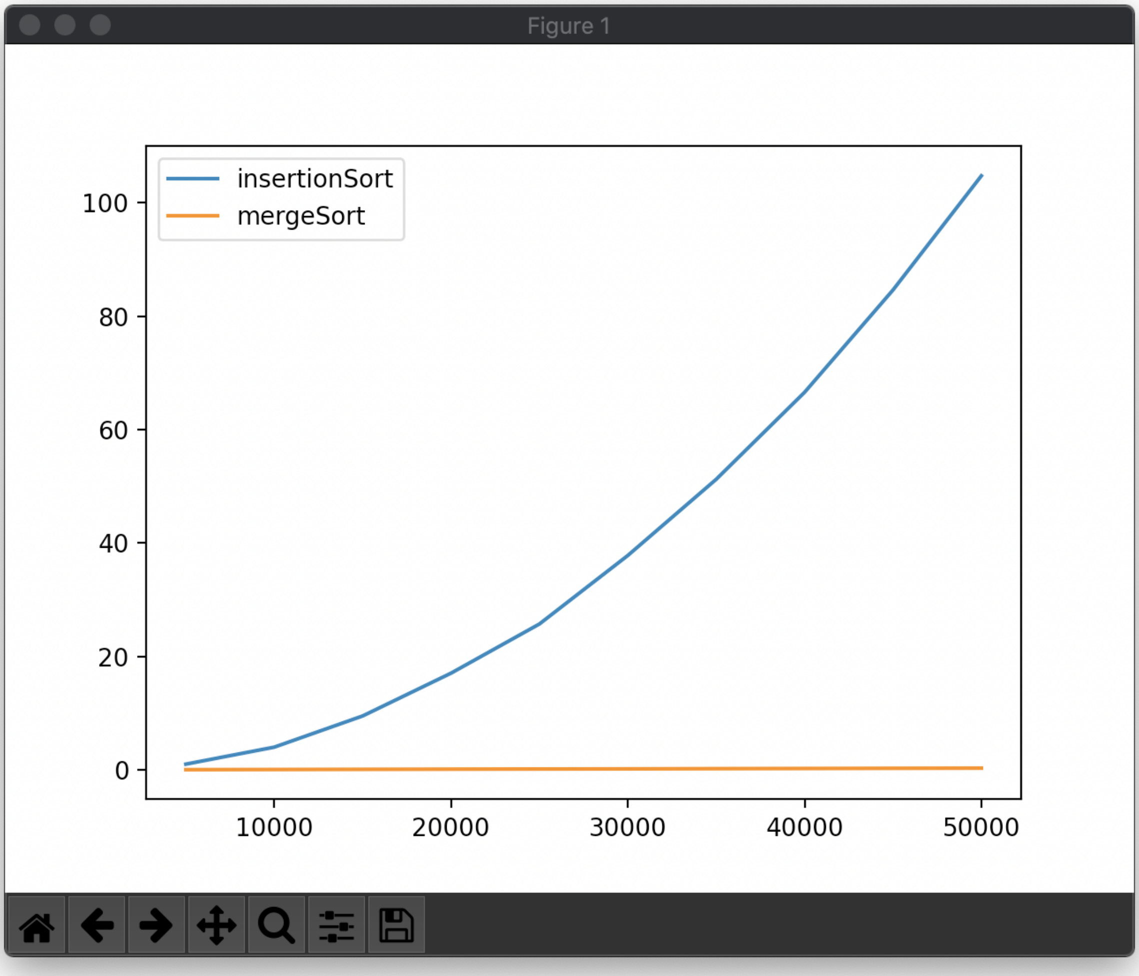


Figure 1
0.30
0.25
0.20
0.15
0.10
0.05
time
10000
20000
30000
40000
50000
n

d) combine



e) Comparison – How does your experimental running times compare to the theoretical running times of the algorithms?

In theoretical merge sort has O(n * log(n)) notation but insertion sort has O(n^2) notation. So, in theory, merge sort is faster than insertion sort. Also, in experimental running times, merge sort is faster than insertion sort as seen above the graph.