CS325 Homework 2 - Lyell Read

Problem 1

A. T(n) = T(n-2) + n. Because this recurrence does not form a tree, it can be considered that it will do n work n/2 times. Therefore, it will produce $n^2/2$ = Theta(n^2).

B. T(n) = $4T(n/2) + n^3$
   A = 4; B = 2; C = 3. $Log_2 4 = 2 < C$ --> Case 3: Theta(n^3)

C. T(n) = $9T(n/3) + n^2$
   A = 9; B = 3; C = 2. $Log_3 9 = 2 = C$ --> Case 2: Theta(n^2Log n)

Problem 2

A. $T(n)_{alg} = 4T(n/3) + n$. A = 4; B = 3; C = 1. $Log_3 4 > C$ --> Theta($n^{log(3,4)}$) ~= Theta($n^{1.2}$)

Problem 3

A.
```
TSA(l, x):
        If len == 1:
             Assert l[0] == x
             Return l[0]

        Leftbreak = len(l)//3
        Rightbreak = leftbreak + (len(l) - leftbreak)//2

        If l[leftbreak] == x:
             Return l[leftbreak]

        Elif l[rightbreak] == x:
             Return l[rightbreak]

        If x < l[leftbreak]:
             Return TSA(l[:leftbreak])

        ElIf l[leftbreak] < x < l[rightbreak]:
             Return TSA(l[leftbreak:rightbreak])

        Else:
             Return TSA(l[rightbreak:])
```

B. T(n) = T(n/3) + c

C. A = 1; B = 3; C = 0; $Log_3 1 = 0 = c$ -->Theta($log_3 n$) by master theorem.

Problem 4

A.
```
Mergesort3 (l):
        If length of l > 1:
```

```
            Leftbreak = len(l)//3
            Rightbreak = leftbreak + (len(l) - leftbreak)//2

            One = l[:leftbreak]
            Two = l[leftbreak:rightbreak]
            Three = l[rightbreak:]

            mergesort3(one)
            mergesort3(two)
            mergesort3(three)

            Output = empty array
            Return merge3(one, two, three, out)

Merge3(a, b, c, o):
      while len(a) > 0 or len(b) > 0 or len(c) > 0:
            if len(a) == 0 and len(b) == 0:
                  Pop c to o
            elif len(a) == 0 and len(c) == 0:
                  Pop b to o
            elif len(b) == 0 and len(c) == 0:
                  Pop a to o
            else:
                  if len(a) == 0:
                        if b[0] > c[0]:
                              Pop c to o
                        else:
                              Pop b to o
                  elif len(b) == 0:
                        if c[0] > a[0]:
                              Pop a to o
                        else:
                              Pop c to o
                  elif len(c) == 0:
                        if    a[0] > b[0]:
                              Pop b to o
                        else:
                              Pop a to o
                  else:
                        t = [a[0], b[0], c[0]]
                        p = min(t)
                        if p == a[0]:
                              Pop a to o
```

```
                    elif p == b[0]:
                        Pop b to o
                    else:
                        Pop c to o
        return o
```

B. T(n) = 3T(n/3) + n

C. A = 3; B = 3; C = 1: $Log_3 3 = 1 = C$ --> Theta(nlogn)

Problem 5

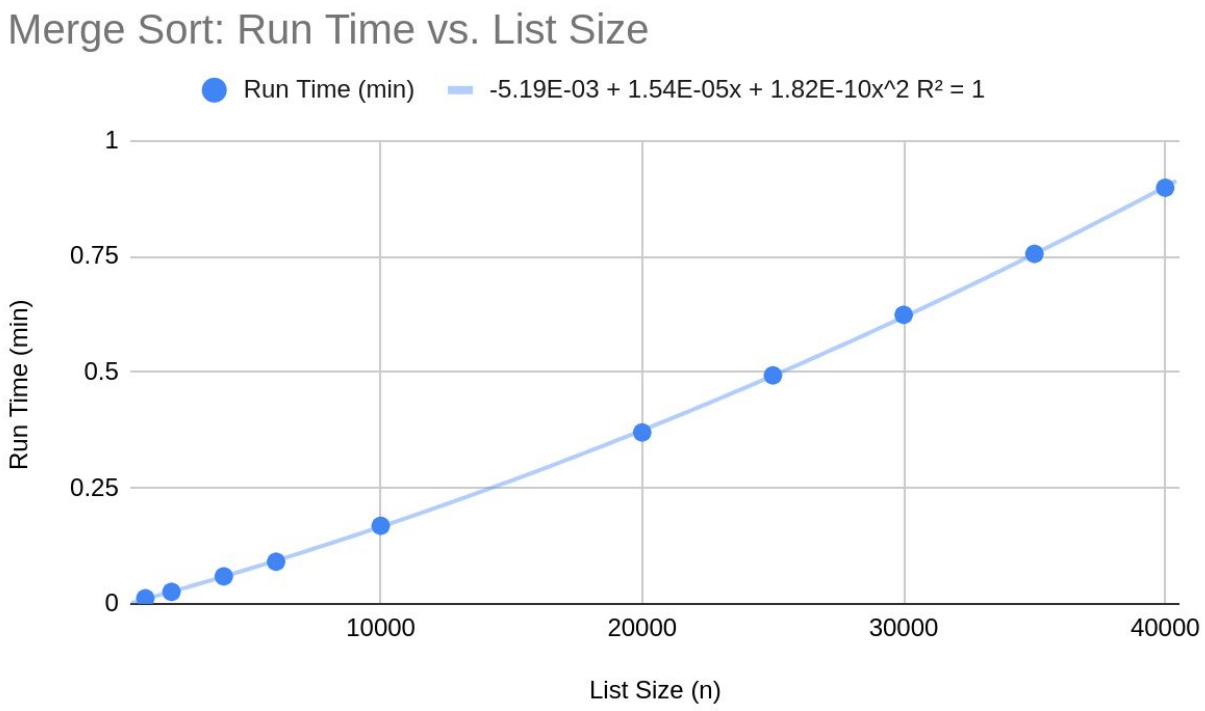
A. Done.

B. Done.

C.

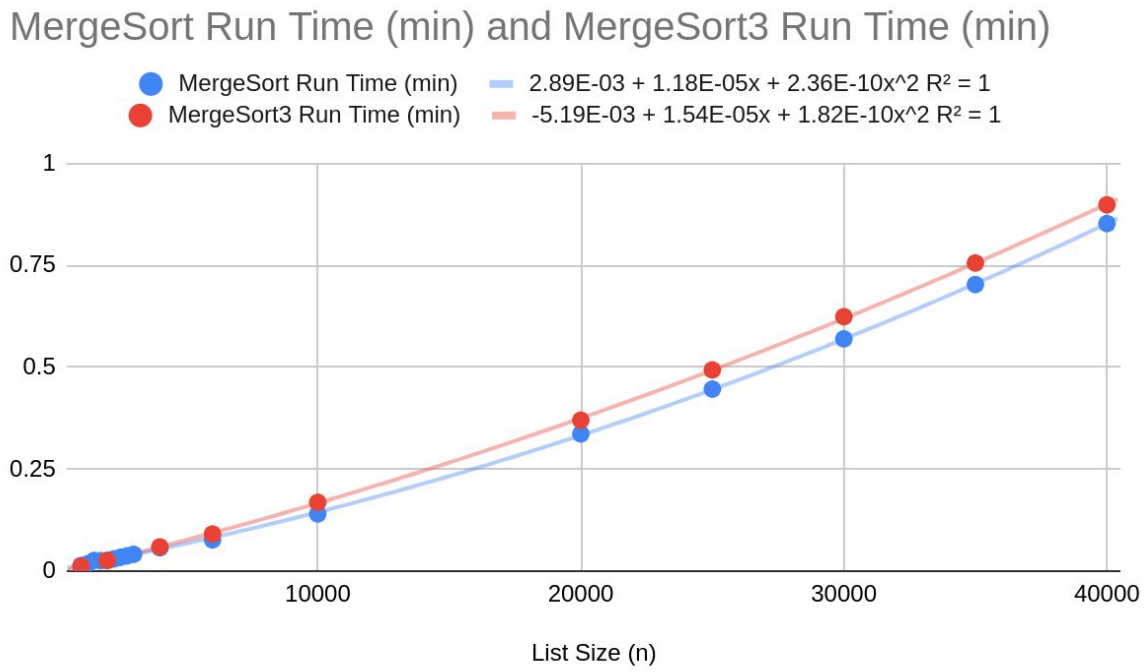
Merge	Run 1	Run 2	Run 3	Run 4	Average
1000	0.01200509071	0.01206111908	0.01221728325	0.01189827919	0.01204544306
2000	0.0259373188	0.02580475807	0.02608108521	0.02573871613	0.02589046955
4000	0.05929946899	0.05958747864	0.05959224701	0.05920624733	0.05942136049
6000	0.09093117714	0.09108948708	0.09163546562	0.09040164948	0.09101444483
10000	0.1687877178	0.1688609123	0.1692454815	0.1674787998	0.1685932279
20000	0.3697695732	0.3712639809	0.3726706505	0.3667051792	0.3701023459
25000	0.4924983978	0.4931645393	0.4969453812	0.4906876087	0.4933239818
30000	0.624707222	0.6235153675	0.6296362877	0.6183440685	0.6240507364
35000	0.7580864429	0.7550749779	0.7576625347	0.7521328926	0.755739212
40000	0.8971865177	0.9017705917	0.9039547443	0.8906011581	0.898378253

D.



The curve that best fits the data is presented in the above graph, and is a polynomial curve, with equation shown.

E.



From this data, it would appear that the mergesort3 sort is slightly faster than the mergesort. This is likely because despite them both being nlogn time, mergesort is $n\log_2 n$, whereas mergesort3 is $n\log_3 n$. Mergesort3 is not excessively faster either as it requires more comparisons than mergesort, and these take time.