Part 1

1.
```
for j in 1 ... n - 1 do
for k in 0 ... j + 2 do
tick
```

   Inner loop: $Hi = j + 2$, $Lo = 0$

   Summation: $\sum_{j=1}^{n-1} j + 3$

$$um_{j=1}^{n-1} j + 3 = \sum_{j=1}^{n-1} j + 3\sum_{j=1}^{n-1} 1 = \frac{n(n-1)}{2} + 3(n-1)$$

2.
```
i ← 1 while i < n do
tick
j ← n
while j  i do
tick
tick
j
i++
```

   Inner Loop: $Hi = n$, $Lo = i$ two ticks in the loop so multiply the hi - low + 1 by 2

   Summation: $\sum_{j=1}^{n-1} 2(n - i + 1) + 1$
   In the summation the low value is one and the high value is n-1 because i started at 1 and went until the less than sign. A one was added for the tick in the outer while loop.

$$\sum_{i=1}^{n-1}(2n - 2i + 2) + 1 = 2n\sum_{i=1}^{n-1} 1 - 2\sum_{i=1}^{n-1} i + 4\sum_{i=1}^{n-1} 1 + 1$$
$$= 2n(n-1) - n(n-1) + 4(n-1) + 1$$

3.
```
tick
i ← 1
while i  n do
tick
j ← 0
while j ¡ i × i do
tick
j++
i++
```

   Inner Loop: $Hi = i^2 - 1$, $Lo = 0$
   It is minus one because their is a less than sign in the inner while loop.

   Summation: $\sum i = 1^n(i^2 - 1 + 1)$
   The sum goes from 1 to n because i starts at 1 and the outer while loop has a less than or equals sign.

Then one is added at the end because their is a tick on the outside.

$\sum i = 1^n i^2 - \sum_{i=1}^{n} 1 + 1$
$\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6} - n + 1$

Part 2

4. Does $(n-2)^2 = \Theta(n \log n)$?

$$f(n) = (n-2)^2, g(n) = n \log n$$

$$\lim_{n\to\infty} \frac{(n-2)^2}{n \log n} = \lim_{n\to\infty} \frac{2(n-1)}{\log n + 1} = \lim_{n\to\infty} 2n = \infty.$$

Since the limit equals infinity as n goes to infinity $(n-2)^2 \neq \Theta(n \log n)$ but $(n-2)^2 = \Omega(n \log n)$.

5. Does $4^{\log_2 n + \log_2 \log_2 n} = \Omega(n^2)$?

$$f(n) = 4^{\log_2 n + \log_2 \log_2 n}, g(n) = n^2$$

$$\lim_{n\to\infty} \frac{4^{\log_2 n + \log_2 \log_2 n}}{n^2} = \lim_{n\to\infty} \frac{n^2 log_2 n^2}{n^2} = \lim_{n\to\infty} log_2 n^2 = \lim_{n\to\infty} 2log_2 n = \infty$$

Since the limit as n approaches infinity equals infinity then the the statement is true that, $4^{\log_2 n + \log_2 \log_2 n} = \Omega(n^2)$.

6. Does $nlog_{10}n = O(nln(n))$?

$$f(n) = n \log n, g(n) = nln(n)$$
$$\lim_{n\to\infty} \frac{n \log n}{n \ln n} = \lim_{n\to\infty} \frac{\log n}{\ln n} = 0.4343$$

Since the limit as n approaches infinity equals a constant of 0.4343, it is greater than zero. This means that $nlog_{10}n \neq O(nln(n))$.

7. Does $(\frac{n^2}{2} - 200n) = \Theta(n^2)$?

$$f(n) = \frac{n^2}{2} - 200n, g(n) = n^2$$
$$\lim_{n\to\infty} \frac{\frac{n^2}{2} - 200n}{n^2} = \lim_{n\to\infty} \frac{\frac{n}{2} - 200}{n} = \lim_{n\to\infty} 1/2 = 1/2$$

Since that limit as n approaches infinity is a positive number equal to 1/2. This means that $(\frac{n^2}{2} - 200n) = \Theta(n^2)$ is true.

8. Does $n \log n = \Omega(n^{\frac{4}{3}})$?

$$f(n) = n \log n, g(n) = n^{\frac{4}{3}}$$
$$\lim_{n\to\infty} \frac{n \log n}{n^{\frac{4}{3}}} = \lim_{n\to\infty} \frac{\log n}{n^{\frac{1}{3}}} = \lim_{n\to\infty} -\frac{1}{n} \frac{1}{3n^{\frac{4}{3}}} = 0$$

Since the limit as n approached infinity equals 0, $n \log n \neq \Omega(n^{\frac{4}{3}})$.

9. Let f(n) and g(n) be non-negative functions of n. Prove using the definitions of O, Ω, and — not using limit tests — that if g(n) = Ω(f(n)), then f(n) + g(n) = $\Theta(g(n))$.

   Starting with the definition for g(n) = Ω(f(n)):

   $$g(n) \geq cf(n)$$

   and the definition for f(n) + g(n) = $\Theta(g(n))$:

   $$c_1 g(n) \leq f(n) + g(n) \leq c_2 g(n)$$

   so if we substitute for f(n) in the second equation it yields:

   $$c_1 g(n) \leq \frac{g(n)}{c} + g(n) \leq c_2 g(n)$$

   $$c_1 g(n) \leq g(n)(\frac{1}{c} + 1) \leq c_2 g(n)$$

   Since C is a random constant, their will always be a constant c_1 less than $\frac{1}{c} + 1$ and a constant c_2 that is greater than $\frac{1}{c} + 1$ that will make inequality (f(n) + g(n) = $\Theta(g(n))$) true.

10. Let f(n) and g(n) be non-negative functions of n. If f(n) = $\Theta(g(n))$, does f(n)/g(n) = Θ(1)? Justify your answer.

    Applying the definition for f(n) = $\Theta(g(n))$ gives:

    $$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

    If this expression is divided bu g(n) it yields:

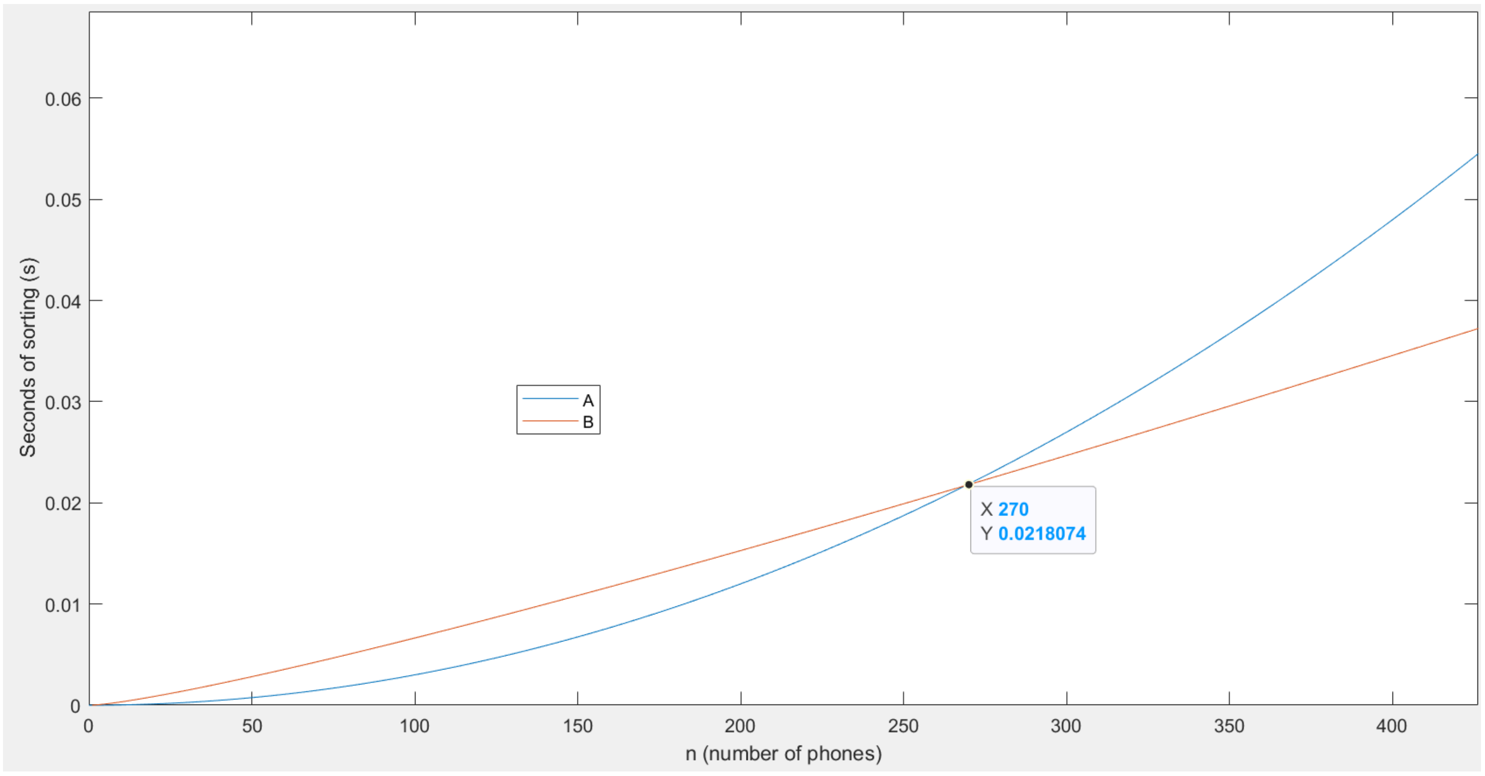
    $$c_1 \leq f(n)(n) \leq c_2$$

    This is the definition of f(n)/g(n) = Θ(1).

Part 3

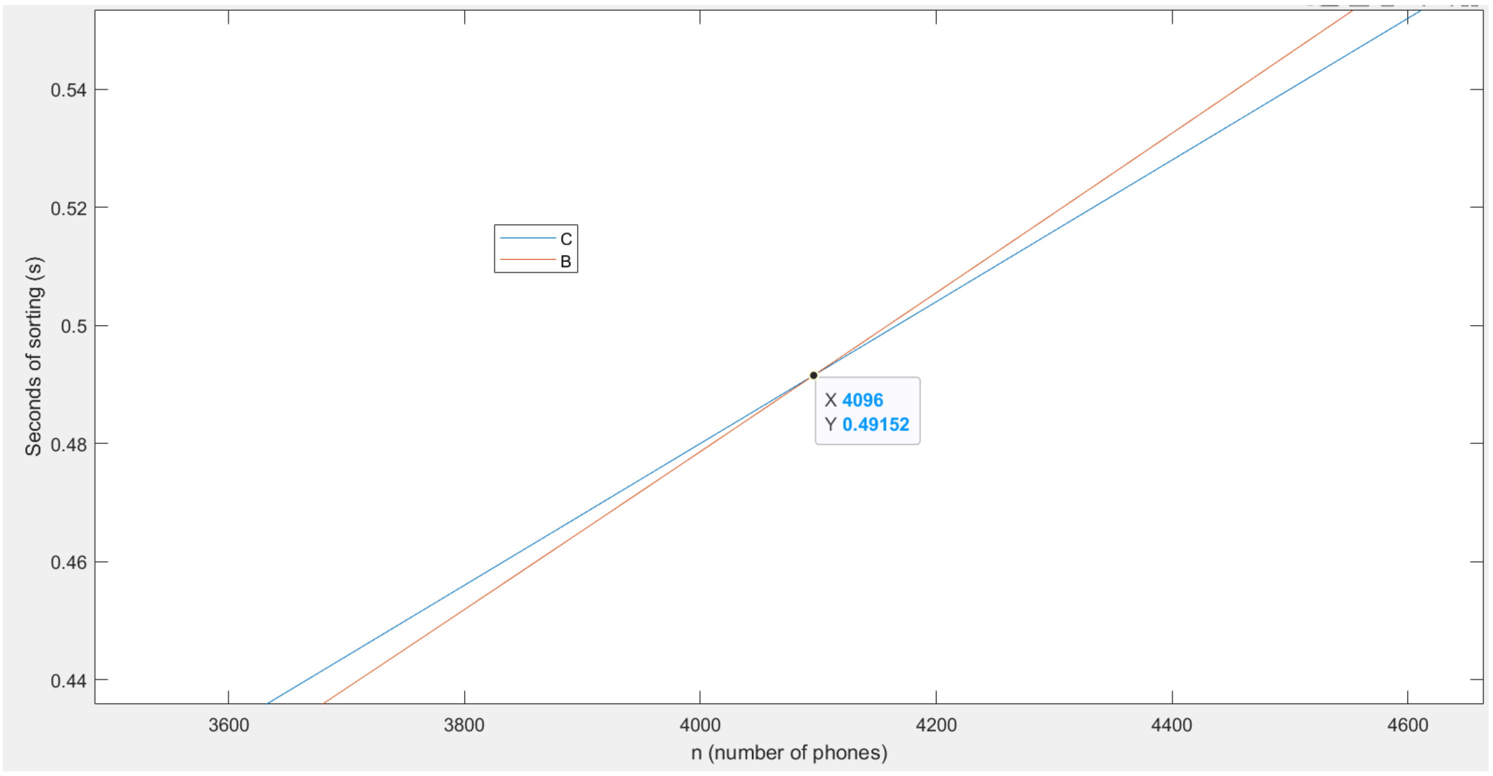
Every night, ATT has to update its database of every customer's telephone number. To enable fast lookups, the database includes an index that is a sorted list of all its phone numbers. This index must be rebuilt each night by re-sorting the contents of the database. ATT has hired you as a consultant to analyze their computing needs for the nightly sorting run. After carefully checking GitHub, you have found three sorting algorithms – A, B, and C – that can sort n phone numbers in $3 * 10^7 n^2, 10^5 nlog_2 n$,and $1.2 * 10^4 n$ seconds respectively on a single processor.

11. (5 pts) What is the smallest problem size n0 such that algorithm B is strictly faster than algorithm A for all $nn0$? (Hint: Plugging in values for n or using Newton's method are acceptable ways to solve the problem, and you are welcome to use online tools such as Wolfram Alpha and Desmos.) Explain your reasoning.



From this graph it can be seen that algorithm B < A when n is greater than 270 phone numbers.

12. (5 pts) What is the smallest problem size n1 for which algorithm C is strictly faster than algorithm B for all n n1? Explain your reasoning.



From this graph that shows the intersection of algorithm's B and C, C will be less than B if n > 4096

13. (6 pts) Describe how to construct a sorting algorithm that always achieves the best running time of any of algorithms A, B, or C for a given n.

    From problems 11 and 12, we found out that B < A when n is greater than 270 and C < B when n is greater than 4096. So to construct a sorting algorithm that would always achieve the best running time of the three algorithms would use if else statements to tell you which algorithm to use for a certain n.

    $$if(n < 270)...chooseA$$

    $$if(n < 4096)...chooseB$$

    $$else...chooseC$$

14. (8 pts) Suppose the phone database contains 108 phone numbers. The updated information arrives at 4 AM, and the company demands that the new database be ready for use by 5 AM, i.e. one hour later. To meet this deadline, you may split the database evenly across k processors and sort each part separately. (For this problem, ignore the cost of putting the pieces back together.) How many processors do you need to meet your deadline with each of algorithms A, B, and C?

    So the number of phones is known to be 10^8 and we also know that the time it needs to take to sort the numbers is 1 hour or less (3600 seconds). So to solve for the amount of processors needed, it should be found how long it takes for the algorithm to sort through all the numbers. For algorithm A:

    $$3600 = 3 * 10^{-7}n^2$$

    $$n = 1.09 * 10^5 phonenumberssortedperhour$$

    $$k = \frac{10^8}{1.09 * 10^5} = 912.87$$

    $$k = 913processors$$

    for algorithm B:

    $$3600 = 10^{-5}n \log n$$

    $$n = 1.509 * 10^7$$

    $$k = \frac{10^8}{1.509 * 10^7} = 6.6$$

    $$k = 7processors$$

    for algorithm C:

    $$3600 = 1.2 * 10^{-4}n$$

    $$n = 3 * 10^7$$

    $$k = \frac{10^8}{3 * 10^7} = 3.33$$

    $$k = 4processors$$