Homework 3

NAME: Kuangyi Yang
ID: A53083212
LOGIN: cs12fig

Part 1A

1. False
2. True
3. False
4. False
5. True
6. True
7. False
8. True
9. False
10. True
11. False
12. False
13. True
14. True
15. False

Part 1B

11. $f(n) = n^2 - 100n - 1000$ is a $O(n^2)$ function, $g(n) = n$ is a $O(n)$ function There is no way to pick a c that would make an $O(n)$ function stay above an $O(n^2)$ function, so in this problem, we only have $f(n) = \Omega(g(n))$, $f(n) = O(g(n))$ is false, so $f(n) = \Theta(g(n))$ is false.

12. $f(n) = n^2 - 4n - 4$ is a $O(n^2)$ function, $g(n) = n^3$ is a $O(n^3)$ function There is no way to pick a c that would make an $O(n^2)$ function stay above an $O(n^3)$ function, so in this problem, we only have $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$ is false, so $f(n) = \Theta(g(n))$ is false.

13. let $f(n) = 100n + \sqrt{n} + 4$ and $g(n) = n$, we can find $c = 101$ and $n_0 = 6$, such that $f(n) \le cg(n)$ for $n \ge n_0$, so $f(n) = O(g(n))$.
We can also find $c = 1$ and $n_0 = 1$,such that $f(n) \ge cg(n)$ for $n \ge n_0$, so $f(n) = \Omega(g(n))$.
Since $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, then $f(n) = \Theta(g(n))$

14. let $f(n) = 2n * n + 10n + 100$ and $g(n) = n^2$, we can find $c = 3$ and $n_0 = 17$, such that $f(n) \le cg(n)$ for $n \ge n_0$, so $f(n) = O(g(n))$.
We can also find $c = 1$ and $n_0 = 1$,such that $f(n) \ge cg(n)$ for $n \ge n_0$, so $f(n) = \Omega(g(n))$.
Since $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, then $f(n) = \Theta(g(n))$

15. $f(n) = n^2 + log(n) * n^2$ is a $O(n^2 log(n))$ function, $g(n) = n^2$ is a $O(n^2)$ function There is no way to pick a c that would make an $O(n^2)$ function stay above an $O(n^2 log(n))$ function, so in this problem, we only have $f(n) = \Omega(g(n))$, $f(n) = O(g(n))$ is false, so $f(n) = \Theta(g(n))$ is false.

Part 2

1) Running time: $O(n)$

Explanation: There is a single loop that runs (2*n-1+1) times. Each time the loop runs it executes 1 instruction, so the goal number of instructions executed is $1 * (2 * n - 1 + 1) = O(2n) = O(n)$

2) Running time: $O(n^2)$

Explanation: There is a single loop that runs $(n^2 - 1 + 1)$ times. Each time the loop runs it executes 1 instruction, so the goal number of instructions executed is $1 * (n^2 - 1 + 1) = O(n^2)$

3) Running time: $O(n)$

Explanation: There is a single loop that runs $(n - 1 + 1)$ times. Each time the loop runs it executes 1 instruction, so the goal number of instructions executed is $1 * (n - 1 + 1) = O(n)$

4) Running time: $O(n^2)$

Explanation: There is an outer loop that runs n times. Each time the loop runs it executes a inner loop, each inner loop runs i times, $i = 1, 2, ..., n$, each time the inner loop runs it executes 1 instruction. so the goal number of instructions executed is $1 * (1 + 2 + ... + n) = O(\frac{1}{2}n^2 + \frac{1}{2}) = O(n^2)$

5) Running time: $O(n)$

Explanation: There is an outer loop that runs 100 times. Each time the loop runs it executes a inner loop, each inner loop runs n times, each time the inner loop runs it executes 1 instruction. so the goal number of instructions executed is $1 * 100 * n = O(100n) = O(n)$

6) Running time: $O(nlog_2n)$

Explanation: There is an outer loop that runs n times. Each time the loop runs it executes a inner loop, each inner loop runs log_2n times, each time the inner loop runs it executes 1 instruction. so the goal number of instructions executed is $1 * n * log_2n = O(nlog_2n)$

7) Running time: $O(n)$

Explanation: There are two loops without nested, the first loop runs n times. Each time the loop runs it executes 1 instruction, the second loop runs $2n + 1$ times, each time the loop runs it executes 1 instruction. so the goal number of instructions executed is $1 * n + 1 * (2n + 1) = O(3n + 1) = O(n)$

8) Running time: $O(1)$

Explanation: There is a single loop that runs 500 times. Each time the loop runs it executes 1 instruction, so the goal number of instructions executed is $1 * 500 = O(500) = O(1)$

9) Running time: $O(n)$

Explanation: There is a single loop that runs n times. Each time the loop runs it executes 3 instruction, so the goal number of instructions executed is $3 * n = O(3n) = O(n)$

10) Running time: $O(n^2)$

Explanation: There are two loops, the first outer loop contains a inner loop. The first loop runs n times. Each time the loop runs it executes a inner loop, each inner loop runs n times, each time the inner loop runs it executes 1 instruction. The second loop runs n times, each time the loop runs it executes 1 instruction. so the goal number of instructions executed is

$$1 * n * n + 1 * n = O(n^2 + n) = O(n^2)$$

Part 3

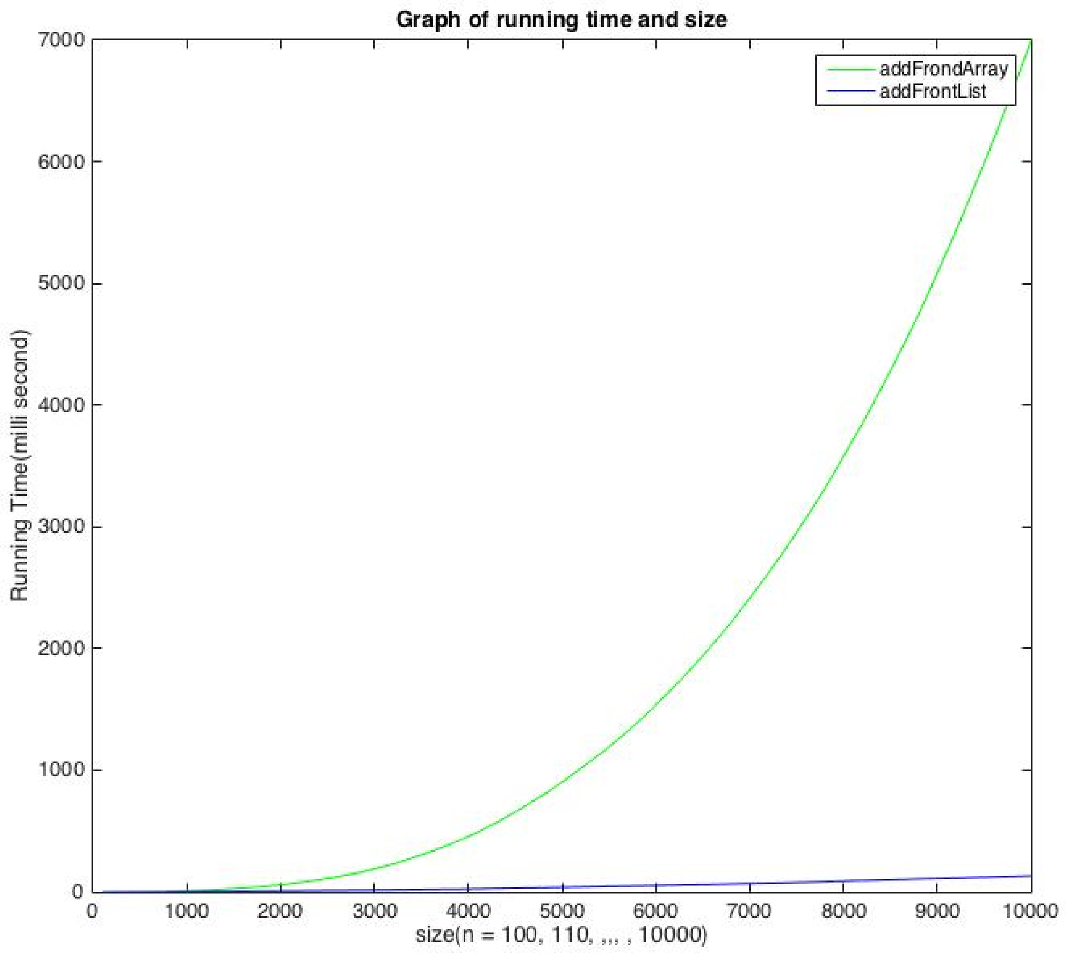

Figure 1: Graph of Running time and Size

Based on the experiments, running time of addFrontArray is quadratic to size of array n, so time complexity of method addFrontArray is $O(n^2)$; This is because when we add n numbers to the front of the array, one by one, there is an outer loop, and this outer loop runs n times, when we add each number to the front of the array, we need to shift the rest numbers in the array, there is an inner loop, and this inner loop runs $n - 1$ times for shifting, so complexity is $n * (n - 1) = O(n^2)$.

Based on the experiments, running time of addFrontList is linear to size of list n, so time complexity of method addFrontArray is $O(n)$; This is because when we add n numbers to the from of the list, one by one, there is a loop, which runs n times. Each time the loop runs it executes 1 instruction to add the from element. So the goal number of instructions executed is $1 * n = O(n)$.