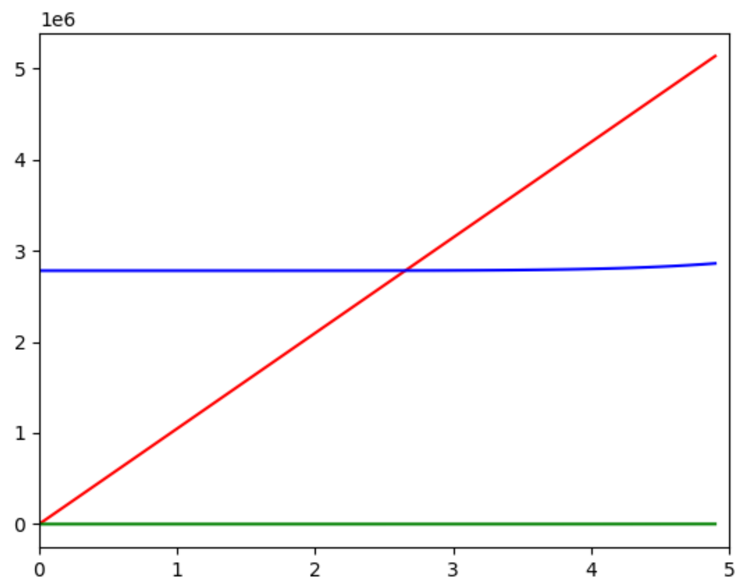
Assignment: Assignment 4
Description: Calculation of Big O and Cost
Student Name: Abhinav Achuta
Student Eid: aa85934
Partner name: Seowon Jeong
Partner Eid: sj32632
Course Number: CS 313E
Unique Number: 52590
Date Created: 09/23/23
Last Modified: 09/27/23

1)

For all the graphs, f1 is red, f2 is blue, and f3 is green

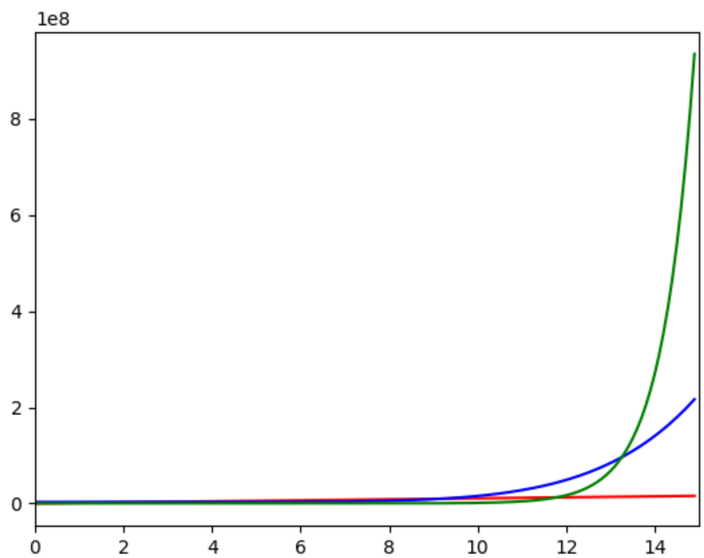
Graph 1: f1(n), f2(n), and f3(n) with n maximum to 5.



Code:

```python
import math
import numpy as np
import matplotlib.pyplot as plt

msize = 5

t = np.arange(0, msize, 0.1)

# red dashes, blue squares and green triangles
# plt.plot(t, t, 'r--', t, t**3.5 - 2**10, 'bs', t, 100*t**2.1 + 50, 'g^')

plt.plot(t, (2**20)*t+2 , 'red', t, t**7.1 + 2.1**20, 'blue', t, 4**t - 2.1**8, 'green')

plt.xlim(0, msize)
plt.rcParams["figure.figsize"] = (7,7)
plt.show()
```

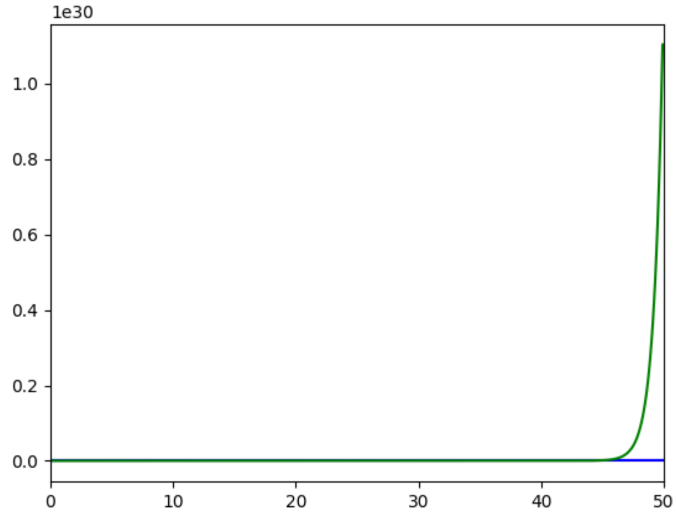
Graph 2: f1(n), f2(n), and f3(n) with n maximum to 15.



Code:

```python
import math
import numpy as np
import matplotlib.pyplot as plt

msize = 15

t = np.arange(0, msize, 0.1)

# red dashes, blue squares and green triangles
# plt.plot(t, t, 'r--', t, t**3.5 - 2**10, 'bs', t, 100*t**2.1 + 50, 'g^')

plt.plot(t, (2**20)*t+2 , 'red', t, t**7.1 + 2.1**20, 'blue', t, 4**t - 2.1**8, 'green')

plt.xlim(0, msize)
plt.rcParams["figure.figsize"] = (7,7)
plt.show()
```

Graph 3: f1(n), f2(n), and f3(n) with n maximum to 50.



Code:

```
import math
import numpy as np
import matplotlib.pyplot as plt

msize = 50

t = np.arange(0, msize, 0.1)

# red dashes, blue squares and green triangles
# plt.plot(t, t, 'r--', t, t**3.5 - 2**10, 'bs', t, 100*t**2.1 + 50, 'g^')

plt.plot(t, (2**20)*t+2 , 'red', t, t**7.1 + 2.1**20, 'blue', t, 4**t - 2.1**8, 'green')

plt.xlim(0, msize)
plt.rcParams["figure.figsize"] = (7,7)
plt.show()
```

<u>Description:</u>

Based on Graph 1 visualizaiton, f1 is increasing linearly, and f2 and f3 show relatively flat curve with little changes. However, according to the function equations, the function f2 and f3 are increasing, but little relative to f1. Based on the Graph 2 visuzaliztion, f2 and f3 starts to increase rapidly starting at approximately n = 12. Especially, f3 is increasing rapidly that f1 seem to be flat curve, and f2 increasing slightly. However, still, based on the function equations, f1 is still increasing and f2 is also increasing rapidly. Based on Graph 3 visualizaiton, f2 function look flat and f1 function is invisible as f3 is increasing too rapidly compared to two functions.

2)

- $f(n) = 2^{(2n+2.3)}$ and $cg(n) = O(2^n)$ - No
  - $2^{(2n+2.3)} <= c(2^n)$ to satisfy
  - If we divide both sides by 2n
  - $2^{(n+2.3)} <= c$, which can not be true
  - As the c is constant and $2^{(n+2.3)}$ is an exponential growing function, big O is not satisfied
- $f(n) = 3^{(2n)}$ and $cg(n) = O(3^n)$ - No
  - $3^{(2n)} <= c(3^n)$ to satisfy
  - If we divide both sides by 3^n
  - $3^n <= c$, which can not be true
  - As the c is constant and 3^n is an exponential growing function, big O is not satisfied

3)

1. **Answer:** f(n) != O(g(n))
   Reasoning:
   When given the equations:
   $$f(n) = (4 \times n)^{150} + (2 \times n + 1024)^{400} \text{ vs. } g(n) = 20 \times n^{300} + (n + 121)^{152}$$
   We can start by taking the dominant terms of f(x) and g(x). The dominant terms of these functions are n^400 and n^300 respectively. Intuitively, we can already see that the term n^400 is much bigger than the term n^300 and can therefore deduce that there is no C to where O(g(n)) is equal to f(n). To prove this further, however, we can input the dominant terms of both functions into the limit definition of the Big O.
   $$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{n^{400}}{n^{300}} = \lim_{n \to \infty} n^{100} = \infty$$
   f(n) = O(g(x)) is only true if this equation does not approach infinity whereas it does approach infinity proven by the limit definition. Therefore, f(n) != O(g(n))
2. **Answer:** f(n) != O(g(n))
   Reasoning:
   When given the equations:
   $$f(n) = n^{1.4} \times 4^{2n} \text{ vs. } g(n) = n^{100} \times 3.99^{n}$$
   We can start by taking the dominant terms of f(x) and g(x). The dominant terms of these functions are 4^2n and 3.99^n respectively. Just by looking at the bases of these two dominant terms, we can already tell that the f(n) function has a bigger value because 4>3. However, we can further look at the exponents which show 2n > n, which again shows that the dominant term for f(n) is greater than that of g(n). Therefore, g(n) will never be greater than or equal to f(n). Which proves that f(n) != O(g(n)).
3. **Answer:** f(n) = O(g(n))
   Reasoning:
   When given the equations:
   $$f(n) = 2^{log_2^n} \text{ vs. } g(n) = n^{1024}$$
   We can start by seeing what the two two terms would equal when n > 1. If n = 2, then f(n) = 1 and g(n) = a value much larger than 1. Therefore, it is already shown that O(g(n)) will always be > f(n) if n >= 1 and c = 1. This proves that f(n) = O(g(n)).

4)

Algorithm 1 What is the Big O of this pseudocode?

```
1:  i = 1
2:  while i ≤ n do
3:      A[i] = i
4:      i = i + 1
5:  end while
6:  for j ← 1 to n do
7:      i = j
8:      while i ≤ n do
9:          A[i] = i
10:         i = i + j
11:     end while
12: end for
```

1. C1 x (n)
2. C2 x (n)
3. ^
4. ^
5. NO COST
6. C6 x (n)
7. C7 x (n - 1)
8. C8 x (n)
9. ^
10. ^
11. NO COST
12. NO COST

For the first loop (2) Big O = O(n)

For the second loop (6) Big O = O(n)
Inside the second loop is a another loop (8) Big O = O(n)
The total for this loop will be O(n) * O(n) = O(n^2)

Because the second loop has the dominant term, the overall Big O = O(n^2).

Answer: O(n^2)

5)

Algorithm 2 What is the Big O of this pseudocode?

```
1: x = 0
2: for i ← 0 to n do
3:     for j ← 0 to (i × n) do
4:         x = x + 10
5:     end for
6: end for
```

2: C1 x (n)
3: C2 x (n x i)

$$\sum_{i=0}^{n} \sum_{j=0}^{i \times n} 1 = \sum_{i=0}^{n} (i \times n + 1)$$

$$= n \times \sum_{i=0}^{n} i + \sum_{i=0}^{n} 1$$

$$= n \times \frac{n(n-1)}{2} + (n+1)$$

$$= \frac{n^3}{2} - \frac{n^2}{2} + n + 1$$

$$\boxed{O(n^3)}$$