MET CS 566 - Analysis of Algorithms

Assignment 1 - 20 Points

Please note the honor code policy regarding this Assignment. You may not discuss particular questions or discuss or transmit answers from the assignment with other people, except for the MET CS 566 teaching team.
Note: You can take a screenshot of your code or a photo of your handwriting answers and insert them into the document. Please save your document as a pdf file and upload it to Blackboard

Tasks

1. Use python to create 3 different plots of the following functions **(4 points)**:

$f_1(n) = (2^{10})(n) + 2^{10}$

$f_2(n) = n^{3.5} - 1000$

$f_3(n) = 100n^{2.1} + 50$

- Create 3 plots and limit the horizontal x-axis to $n = 5, 15, 50$. On each of the 3 plots, you need to show the above 3 functions. On the first plot, the x-axis is limited to 5, on the second one x-axis is limited to 15 and on the 3rd one x-axis is limited to 50
- Visualize the 3 functions in 3 colors (f_1 in red, f_2 in blue, f_3 in green)
- Describe your visualization and what you see in these 3 plots.
- Add your visualization and your python code to your PDF report file

2. Asymptotic Notation **(3 points)**

• Is $2^{(n+1.3)} = O(2^n)$?

• Is $3^{(2\times n)} = O(3^n)$?

Please first give your conclusion (e.g., 'yes' or 'no') then describe your answer.

3. For each pair of functions $f(n)$ and $g(n)$, check if $f(n) = \theta(g(n))$?

Functions $f(n)$ and $g(n)$ are:

1. $f(n) = (4 \times n)^{150} + (2 \times n + 1024)^{400}$ $vs.$ $g(n) = 20 \times n^{400} + (n + 1024)^{200}$
2. $f(n) = n^{1.4} \times 4^n$ $vs.$ $g(n) = n^{200} \times 3.99^n$
3. $f(n) = 2^{\log(n)}$ $vs.$ $g(n) = n^{1024}$

Please first give your conclusion then describe your answer. You do not need to provide formal proofs. **(6 points)**

4. Analyze the Algorithm 1 and give a tight θ bound on the running time as a function of n. Carefully describe your justifications **(4 points)**

Algorithm 1 What is the tight Θ 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
```

5. Analyze the Algorithm 2. What is the Big O on the running time as a function of n. Carefully describe your justifications. **(3 points)**

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
```