CS566 Assignment 1

Yiduo Feng
09/16/2022

Tasks

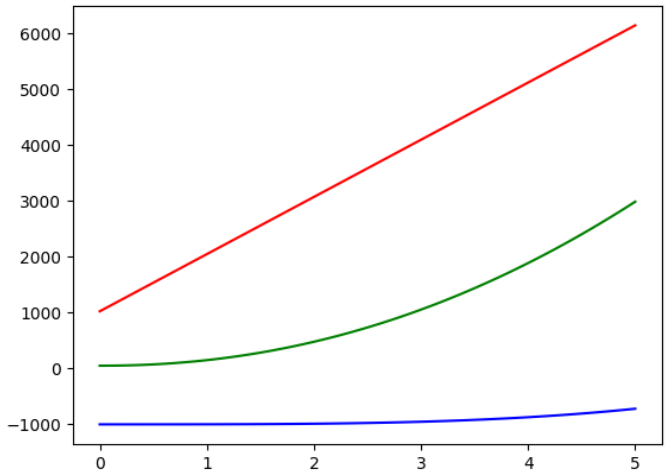
1. Use python to create 3 different plots of the following functions
$f1(n)$ = (2^10)(n) + 2^10
$f2(n)$ = n^3.5 – 1000
$f3(n)$ = 100n^2.1 + 50

- 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 xaxis is limited to 15 and on the 3rd one x-axis is limited to 50
- Visualize the 3 functions in 3 colors ($f1$ in red, $f2$ in blue, $f3$ 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

The code is shown below.

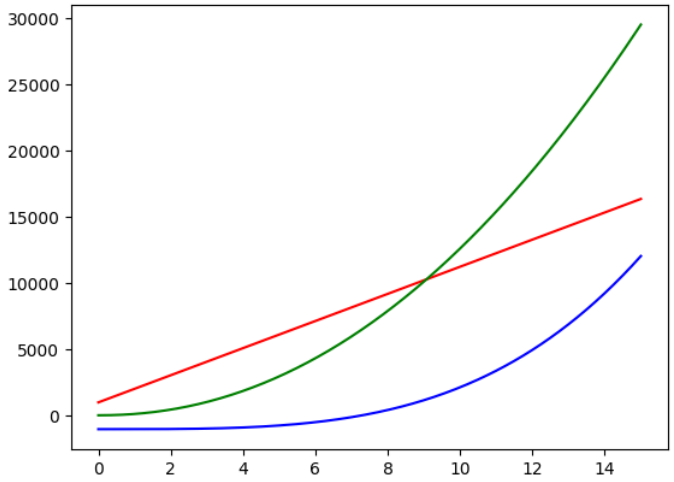
```
import matplotlib.pyplot as plt
import numpy as np

def tesk1(limit, Numberofptr):
    n = np.linspace(0, limit, Numberofptr)
    f_1 = pow(2, 10)*n + pow(2, 10)
    f_2 = pow(n, 3.5) -1000
    f_3 = 100*pow(n, 2.1) + 50

    plt.plot(n, f_1, color='red')
    plt.plot(n, f_2, color='blue')
    plt.plot(n, f_3, color='green')
    plt.show()

tesk1(5, 100)
tesk1(15, 300)
tesk1(50, 1000)
```

- the x-axis is limited to 5:



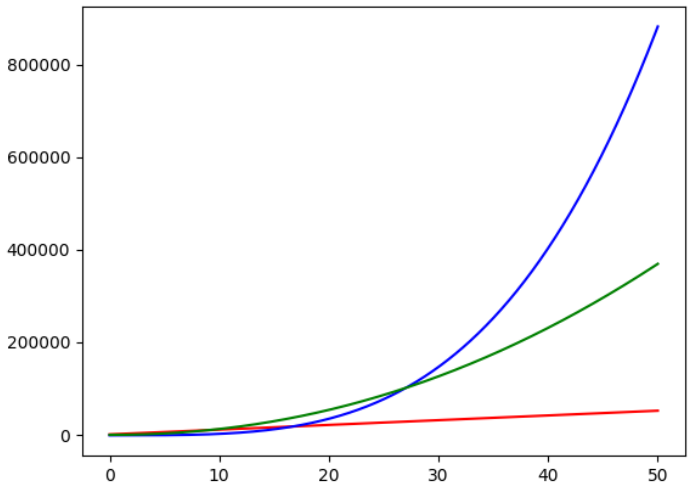
When the x-axis is limited to 5, the blue one is lowest so that it's fastest, and red one is highest so that it's slowest.

- the x-axis is limited to 15



When the x-axis is limited to 15, the plot changes by comparing with that when x is limited to 5. The blue one is still lowest so that it's fastest, and green one is highest so that it's slowest.

- the x-axis is limited to 50



When the x-axis is limited to 50, the plot changes by comparing with that when x is limited to 5 and 15. The red one is lowest so that it's fastest, and blue one is highest so that it's slowest.

Similar to the example in class, the plot changes by changing the limit of x-axis, so when we want to know which expression is fast we need to use the number as large as possible.

2. Asymptotic Notation (3 points)

- Is 2^(n+1.3) = O(2^n) ?

Yes.

Because for c= 3, 3*2^n > 2^(n+1.3) when n > 1

- Is 3^(2×n) = O(3^n)?

No.

Because for c is any number, c*3^n < 3^(2n3) when n > 1

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 × n)^150 + (2 × n + 1024)^400 $vs.$ $g(n)$ = 20 × n^400 + (n + 1024)^200
   Yes.
   When we try to simplify the expression (4 × n)^150 + (2 × n + 1024)^400 , we will get a expression with degree of 400. It's roughly is 4^150*n^150 + (2n)^400 + some other low degree part. According to the example proved in class, we can know that f(n) = θ(n^400). In the same way, g(n) has the highest degree at 400, so g(n) = θ(n^400). Therefore, f(n) = θ(g(n)).

2. f (n) = n^1.4 × 4^n $vs.$ $g(n)$ = n^200 × 3.99^n
   No.
   Since n^1.4 != n^200

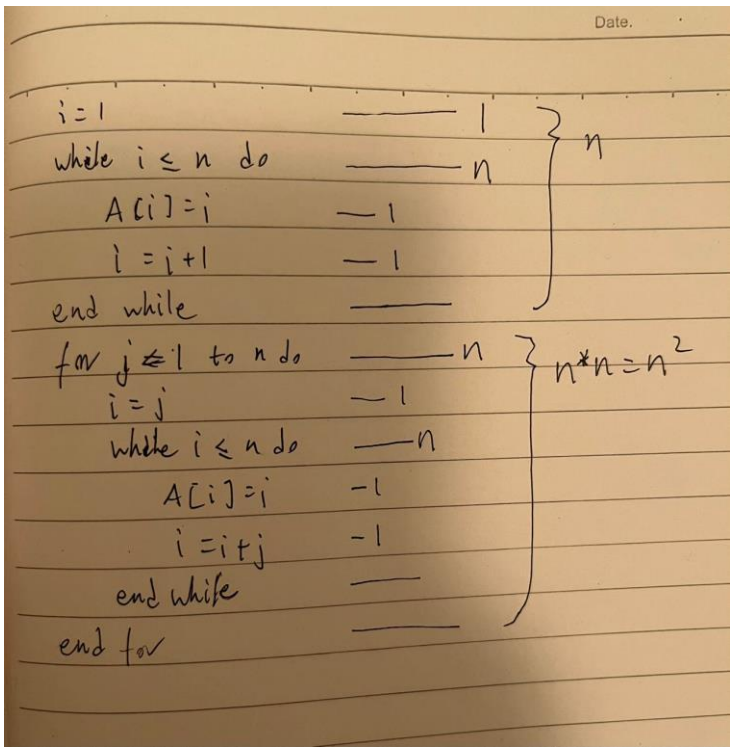
4^n != 3.9^n
Hence, F(n) = θ(n^1.4 × 4^n) != θ(g(n))

3. f (n) = 2^log (n) $vs.$ $g(n)$ = n^1024
   No.
   2^log (n) = n^log(2)
   Log(2) = 0.3 < 1024
   So f(n) = θ(n^0.3) != θ(g(n))

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
```

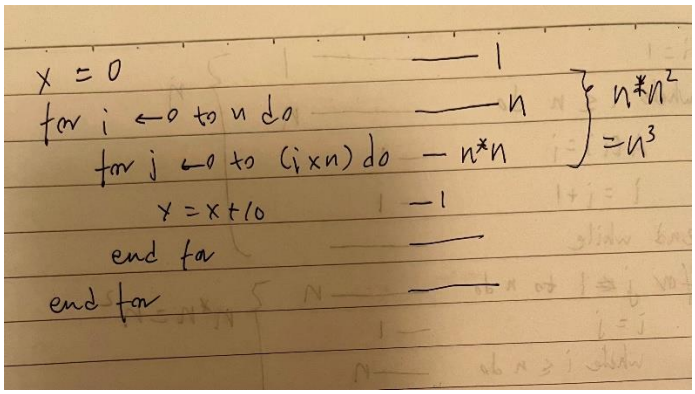


Therefore, n^2+n+6 = θ(n^2)

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
```



Therefore, n^3+2 = O(n^3)