aula1-teorico

August 13, 2024

1 CLASS 1: Theoretical exercise on complexity analysis

1.0.1 1) Solve the following questions, use the asymptotic analysis definition and plot comparative graphs of the functions:

a) Show that the function $5n^2+n$ is $O(n^2)$. To do this, show that there exist two positive constants c and m such that $g(n) \leq cf(n)$, for all $n \geq m$. Solution (a): * The functions are as follows: - $g(n) = 5n^2 + n$ - $f(n) = n^2$ * To say that the function $f(n)$ dominates asymptotically the function $g(n)$ we have to find two positive constants c and m such that $5n^2 + n \leq c * n^2$ for all $n \geq m$

$$5n^2 + n \leq c * n^2 \tag{1}$$

From equation (1) we move everything to one side

$$0 \leq c * n^2 - 5n^2 - n \tag{2}$$

From equation (2) we factor as much as possible

$$((c-5) * n - 1)n \geq 0 \tag{3}$$

From equation (3) we have two equations

$$(c-5) * n - 1 \geq 0 \quad \& \quad n > 0 \tag{4}$$

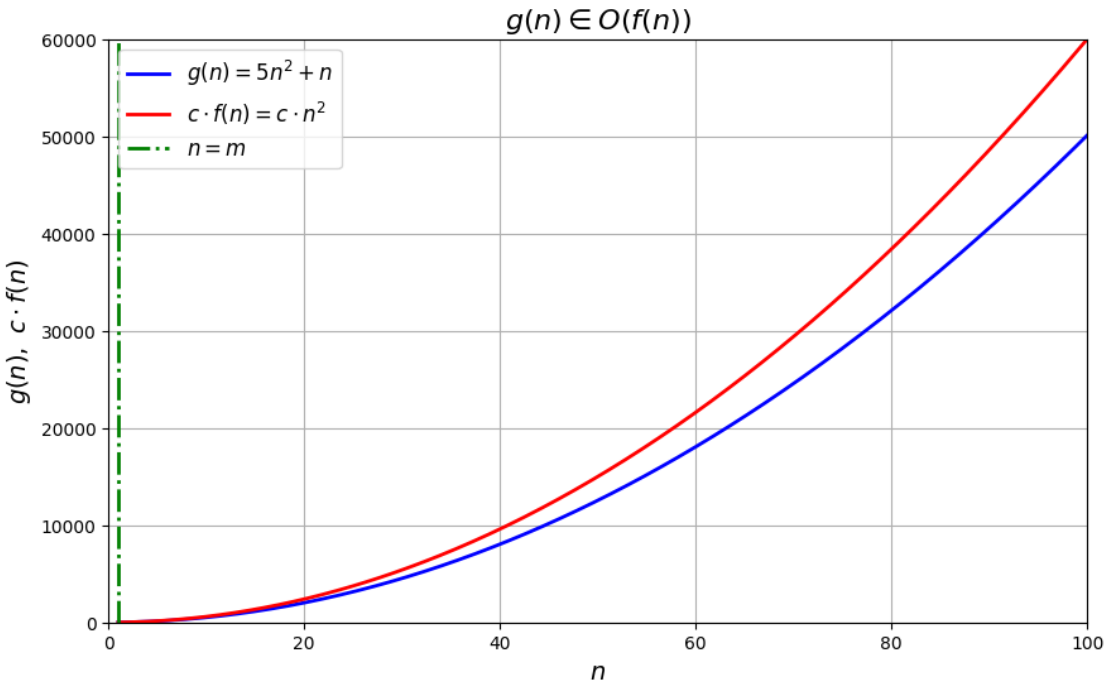
We work only with the first equation

$$(c-5) \geq \frac{1}{n} \tag{5}$$

We start testing numbers, if we set that $n = 1$ it is enough that $c = 6$ for the inequality to satisfy, then we could choose that $c = 6$ and $m = 1$ and the final equation would remain

$$5n^2 + n \leq 6 * n^2 \quad \text{for all} \quad n \geq 1 \tag{end}$$

[3]:
```python
# Python version 3.10.12
# Chatgpt helped me graph the function in python
import matplotlib.pyplot as plt
import numpy as np

# Define the functions
def g(n):
    return 5*n**2 + n

def f(n):
    return n**2

# Sets values found in the previous solve for c and m
c = 6
m = 1

# Generate values for n
n_values = np.linspace(1, 100, 400)

# Draw the functions
plt.figure(figsize=(10, 6))
plt.plot(n_values, g(n_values), label=r'$g(n) = 5n^2 + n$', color='blue',␣
 ↪linewidth=2)
plt.plot(n_values, c * f(n_values), label=r'$c \cdot f(n) = c \cdot n^2$',␣
 ↪color='red', linewidth=2)

# Draw the region where n   m
plt.axvline(x=m, color='green', linestyle='-.', label=r'$n = m$', linewidth=2)

# Add tags and title
plt.xlabel(r'$n$', fontsize=14)
plt.ylabel(r'$g(n), \ c \cdot f(n)$', fontsize=14)
plt.title(r'$g(n) \in O(f(n))$', fontsize=16)
plt.legend(fontsize=12)
plt.grid(True)
plt.ylim([0, 60000])
plt.xlim([0, 100])

# Show the graph
plt.show()
```



b) Show that the function $2n^4 + 2n^2$ is $\Omega(n^4)$. To do this, show that if there exist two constants c and m such that $f(n) \geq cg(n)$, for all $n \geq m$. Solution (b): * To demonstrate that the function $f(n) = 2n^4 + 2n^2$ is $g(n) = n^4$ it must be fulfilled that there exist two constants c and m such that:

$$2n^4 + 2n^2 \geq c * n^4 \quad \text{para todo} \quad n \geq m \tag{1}$$

$$2n^4 + 2n^2 \geq c * n^4 \tag{1}$$

From equation (1) we move everything to one side

$$2n^4 + 2n^2 - c * n^4 \geq 0 \tag{2}$$

From equation (2) we factor as much as possible

$$((2 - c) * n^2 + 2)n^2 \geq 0 \tag{3}$$

From equation (3) we have two equations

$$(2 - c) * n^2 + 2 \geq 0 \quad \& \quad n > 0 \tag{4}$$

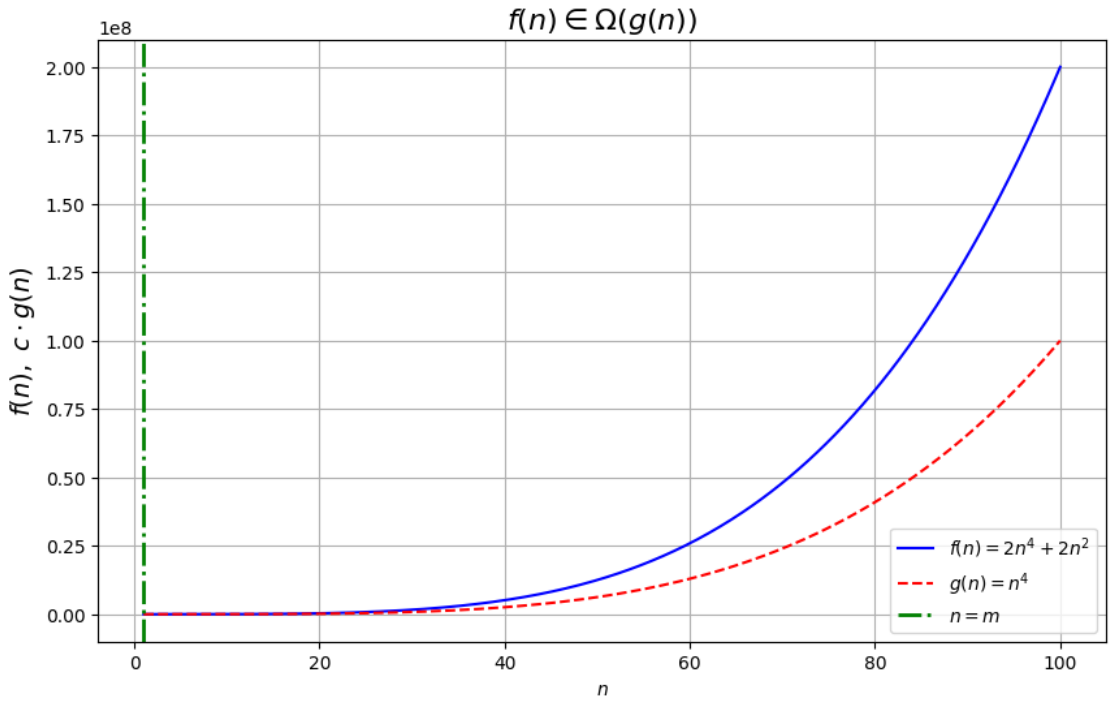
We work only with the first equation

$$(2-c) * n^2 \geq -2 \tag{5}$$

We start testing numbers, if we put that $n = 1$ c could be $1, 2, 3, 4$ to satisfy the inequality, then we could choose that $c = 1$ and $m = 1$ and the final equation would remain

$$2n^4 + 2n^2 \geq 1 * n^4 \quad \text{for all} \quad n \geq 1 \tag{end}$$

[16]:
```python
# Python version 3.10.12
# Chatgpt helped me graph the function in python
import matplotlib.pyplot as plt
import numpy as np

# Definir la función f(n) y g(n)
def f(n):
    return 2 * n**4 + 2 * n**2

def g(n):
    return n**4

# Crear un rango de valores para n
n = np.linspace(1, 100, 400)

# Valor de m
m = 1

# Calcular los valores de f(n) y g(n)
f_n = f(n)
g_n = g(n)

# Graficar f(n) y g(n)
plt.figure(figsize=(10, 6))
plt.plot(n, f_n, label='$f(n) = 2n^4 + 2n^2$', color='blue')
plt.plot(n, g_n, label='$g(n) = n^4$', color='red', linestyle='--')

# Dibuja la región donde n   m
plt.axvline(x=m, color='green', linestyle='-.', label=r'$n = m$', linewidth=2)

# Añadir etiquetas y leyenda
plt.xlabel('$n$')
plt.ylabel(r'$f(n), \ c \cdot g(n)$', fontsize=14)
plt.title(r'$f(n) \in \Omega(g(n))$', fontsize=16)
plt.legend()
plt.grid(True)

# Mostrar la gráfica
plt.show()
```



c) Is it true that $2^{n+1} = O(2^n)$? Is it true that $2^{2n} = O(2^n)$

d) Sort the following functions in order of increasing value. Use function inequality/limits. $n * 2^n$

$(logn)!$

$(n+1)!$

1

2^{2n}

$4n$

$2n * logn$

Solution (d) * Ordered it would be as follows:

1 -> Funcion constante

$4n$ -> Funcion lineal

$2n * logn$ -> Funcion lineal logaritmica

$(logn)!$ -> Funcion factorial

$(n+1)!$ -> Funcion factorial

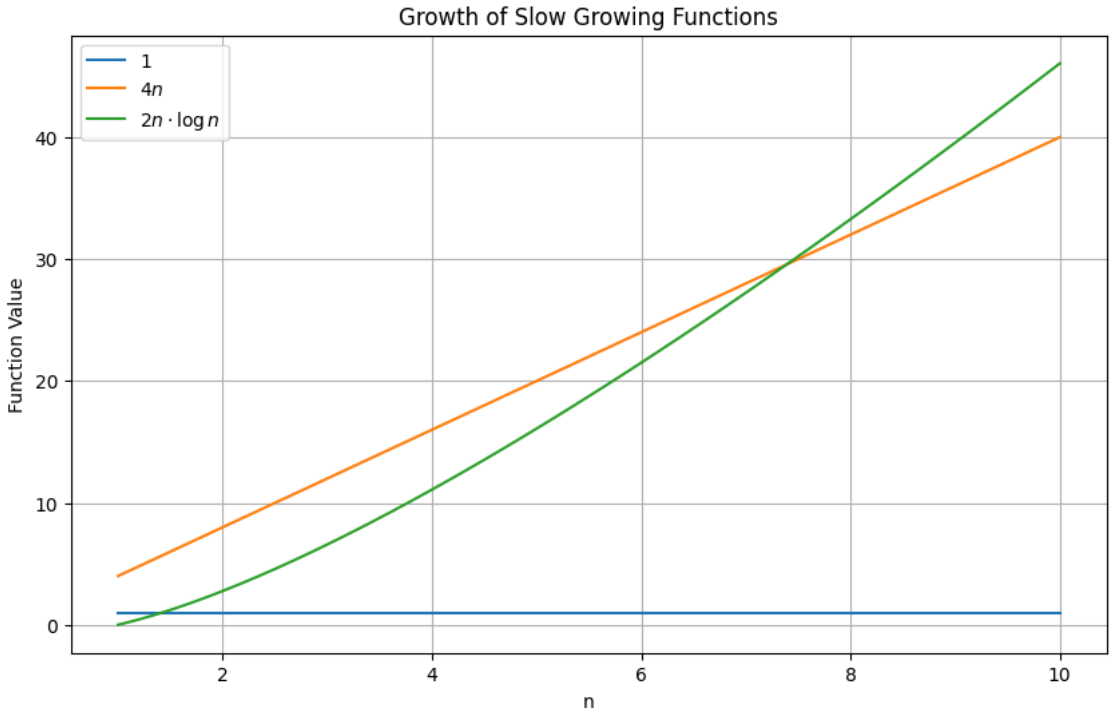
$n * 2^n$ -> Funcion exponencial

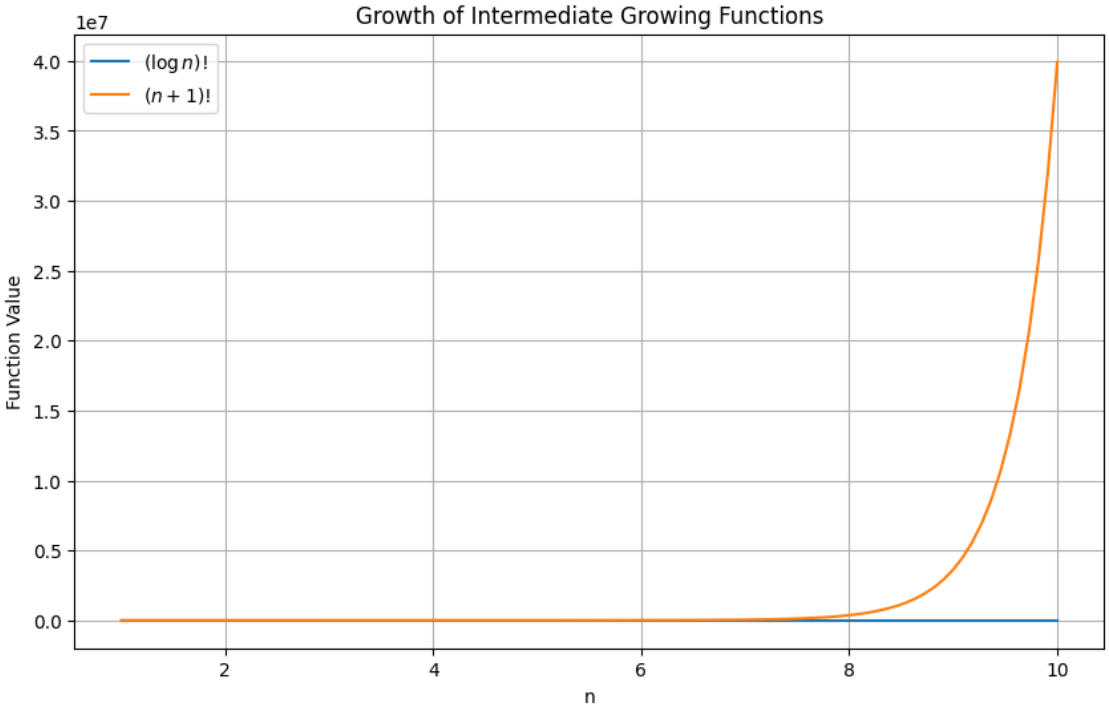
2^{2n} -> Funcion exponencial

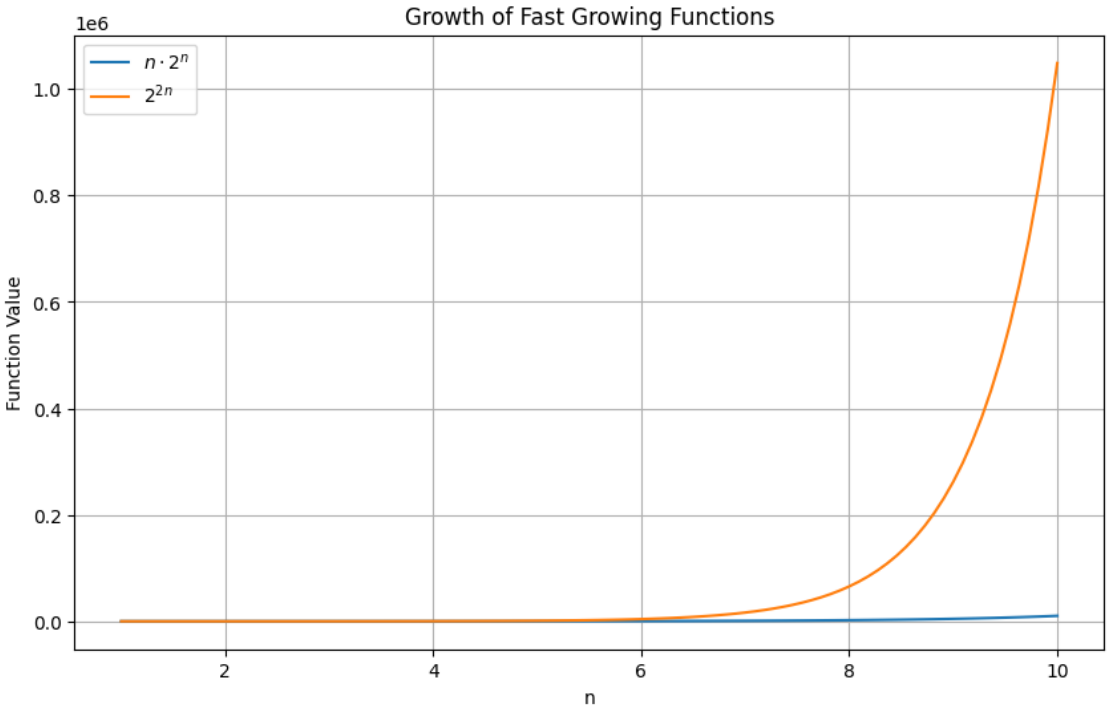
The graphics were made by ChatGPT.

[20]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import factorial

# Define a smaller range of n
n = np.linspace(1, 10, 100)

# Define the functions
f1 = n * 2**n
f2 = factorial(np.log(n))
f3 = factorial(n + 1)
f4 = np.ones_like(n)
f5 = 2**(2*n)
f6 = 4*n
f7 = 2*n*np.log(n)

# Plot the functions in different groups to avoid extreme overlap

# Group 1: Slow growing functions
plt.figure(figsize=(10, 6))
plt.plot(n, f4, label=r'$1$')
plt.plot(n, f6, label=r'$4n$')
plt.plot(n, f7, label=r'$2n \cdot \log n$')

plt.xlabel('n')
plt.ylabel('Function Value')
plt.title('Growth of Slow Growing Functions')
plt.legend()
plt.grid(True)
plt.show()

# Group 2: Intermediate growing functions
plt.figure(figsize=(10, 6))
plt.plot(n, f2, label=r'$(\log n)!$')
plt.plot(n, f3, label=r'$(n+1)!$')

plt.xlabel('n')
plt.ylabel('Function Value')
plt.title('Growth of Intermediate Growing Functions')
plt.legend()
plt.grid(True)
plt.show()

# Group 3: Fast growing functions
```

```
plt.figure(figsize=(10, 6))
plt.plot(n, f1, label=r'$n \cdot 2^n$')
plt.plot(n, f5, label=r'$2^{2n}$')

plt.xlabel('n')
plt.ylabel('Function Value')
plt.title('Growth of Fast Growing Functions')
plt.legend()
plt.grid(True)
plt.show()
```




Growth of Intermediate Growing Functions
1e7
$(\log n)!$
$(n+1)!$
Function Value
n
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
2
4
6
8
10


Growth of Fast Growing Functions
1e6
$n \cdot 2^n$
2^{2n}
Function Value
n
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10

1.0.2 Question 2

Complexity analysis evaluates the efficiency of algorithms in terms of time and space, helping to measure how many resources are needed depending on the size of the input. It is used to compare algorithms and choose the most efficient one to solve a problem.