.010CS325
Chih Hsuan Huang

Problem 1.

Problem 1: indecate whether $f = O(g)$ $f = \Omega(g)$ $f = \Theta(g)$

(a) $f(n) = 12n-5$ $g(n) = 1235813n + 2017$

① determine if $f(n) = O(g(n)) \Rightarrow 12n-5 = O(g(n))$

Find constants c, n_0 s.t. $c, n_0 > 0$. $\forall n \geq n_0$

$c \cdot g(n) \geq f(n)$ $\forall n \geq n_0$

$\Leftrightarrow c \cdot (1235813n + 2017) \geq 12n-5$ $\forall n \geq n_0$

$c = 1, n_0 = 1$: $1235813n + 2017 \geq 12n-5$ $\forall n \geq 1$

$\Leftrightarrow 1235813n \geq 12n$ $\forall n \geq 1$

$\therefore f(n) = O(g(n))$

② determine if $f(n) = \Omega(g(n)) \Rightarrow 12n-5 = O(g(n))$

Find constants c, n_0 s.t. $c, n_0 > 0$ s.t. $\forall n \geq n_0$

$f(n) \geq c \cdot g(n)$

$\Leftrightarrow 12n-5 \geq c \cdot (1235813n + 2017)$ $\forall n \geq n_0$

$c = 10^{-6}$ $n = 1$ $12n-5 \geq 10^{-6} \cdot (1235813n + 2017)$ $\forall n \geq 1$

$\therefore f(n) = \Omega\, O(g(n))$

$\Rightarrow \because f(n) = O(g(n))$ & $f(n) = \Omega(g(n))$

$\therefore f(n) = \Theta(g(n))$

(b) $f(n) = n\log n$, $g(n) = 0.00000001\,n$

① determine if $f(n) = O(g(n))$

find constant c, n_0 s.t. $c, n_0 \geq 0$ $\forall\, n \geq n_0$

$\Leftrightarrow$ $c \cdot (0.00000001\,n) \geq n\log n$ $\forall\, n \geq n_0$

$\Leftrightarrow$ $c \cdot 0.00000001 \geq \log n$ $\forall\, n \geq n_0$

$\because$ $\log n$ is growing faster than $0.00000001c$

$\therefore$ $c \cdot 0.00000001 \leq \log n$ $\forall\, n \geq n_0$

$\Rightarrow f(n) \neq O(g(n))$

② determine if $f(n) = \Omega(g(n))$

$f(n) \geq c \cdot g(n)$ $\forall\, n \geq n_0$

$\because \log n \geq 0.00000001c$ $\forall\, n \geq n_0$

$\therefore f(n) = \Omega(g(n))$

$\Rightarrow$ $\because$ $f(n) \neq O(g(n))$ and $f(n) = \Omega(g(n))$

$\therefore f = \Omega(g)$

(c) $f(n) = n^{\frac{2}{3}}$, $g(n) = 7n^{\frac{3}{4}} + n^{\frac{1}{10}}$

① determine if $f(n) = O(g(n))$

$\Rightarrow$ find the constant c, n_0 s.t. $c, n_0 > 0$ $\forall\, n \geq n_0$

$c \cdot g(n) \geq f(n)$ $\forall\, n \geq n_0$

$\Leftrightarrow$ $c \cdot (7n^{\frac{3}{4}} + n^{\frac{1}{10}}) \geq n^{\frac{2}{3}}$ $\forall\, n \geq n_0$

where $c=1$, $n_0=1$ $7n^{\frac{3}{4}} + n^{\frac{1}{10}} \geq n^{\frac{2}{3}}$ $\forall\, n \geq 1$

$\because n^{\frac{3}{4}}$ will grow faster that $n^{\frac{2}{3}}$

$\therefore n^{\frac{3}{4}} \geq n^{\frac{2}{3}}$

$\Rightarrow f(n) = O(g(n))$

$\Rightarrow$ $\because$ $n^{\frac{3}{4}}$ will only greater that $n^{\frac{2}{3}}$

$\therefore$ $f(n) = O(g(n))$ and $f(n) \neq \Omega(g(n))$

$\Rightarrow f(n) = O(g(n))$.

(d) $f(n) = n^{1.0001}$, $g(n) = n\log n$

$\because$ $n^{1.0001}$ will always grow faster than $n\log n$

$\Rightarrow \lim_{n\to\infty} \frac{n^{1.0001}}{n\log n} = \infty$

$\Rightarrow$ find the constant c, n_0 s.t. $c, n_0 > 0$ $\forall\, n \geq n_0$

$\Rightarrow n\log n \leq n^{1.0001}$ $\forall\, n \geq n_0$

$\Rightarrow c \cdot n\log n \leq n^{1.0001}$ $\forall\, n \geq n_0$

Where $c=1$ $n_0=1$: $n\log n \leq n^{1.0001}$ $\forall\, n \geq 1$

$\Rightarrow$ $f(n) = \Omega(g(n))$ and $f(n) \neq O(g(n))$

$\Rightarrow$ $f = \Omega(g)$

(e) $f(n) = n6^n$, $g(n) = (3^n)^2$

① determine if $f(n) = O(g(n))$

find the constant c, n_0 s.t. $c, n_0 > 0$ $\forall\, n \geq n_0$

$c \cdot g(n) \geq f(n)$ $\forall\, n \geq n_0$

$c \cdot (3^n)^2 \geq n6^n$ $\forall\, n \geq n_0$

Where $c=1$ $n_0=1$: $(3^n)^2 \geq n6^n$ $\forall\, n \geq 1$

$9^n \geq n6^n$ $\forall\, n \geq 1$

$\Rightarrow$ $f(n) = O(g(n))$

② determine if $f(n) = \Omega(g(n))$

$f(n) \geq c \cdot g(n)$

find the constant c, n_0 s.t. $c, n_0 > 0$ $\forall\, n \geq n_0$

$n6^n \geq (3^n)^2$ $\forall\, n \geq n_0$

$n6^n \geq 9^n$ $\forall\, n \geq n_0$

$\Rightarrow$ since 9^n will grow faster than 6^n

$\Rightarrow$ $9^n \geq n6^n$

$\Rightarrow$ $f(n) \neq \Omega(g(n))$ $\Rightarrow$ $f(n) = O(g(n))$

Problem 2

Problem 2. Prove that $\log(n!) = \Theta(n \log n)$

① determine if $\log(n!) = O(n \log n)$

Find the constant c, n_0 s.t. $c, n_0 > 0$ $c \cdot n \log n \geq \log n!$ $\forall\ n \geq n_0$

$$\begin{aligned}
\log(n!) &= \log(1 \cdot 2 \cdot 3 \cdot \cdots (n-1) \cdot n) \\
&= \log 1 + \log 2 + \log 3 + \cdots + \log n \\
&\leq \log(n) + \log(n) + \cdots + \log(n) \\
&= n \log n
\end{aligned}$$

$\because n \log n = \log(n^n) > \log(n!)$

$\therefore$ where $c=1$ $n_0=1$ $n \log n > \log(n!)$ $\forall\ n \geq 1$

$\therefore \log(n!) = O(n \log n)$

② determine if $\log(n!) = \Omega(n \log n)$

Find the constant c, n_0 s.t. $c, n_0 > 0$ $\log n! \geq c \cdot n \log n$ $\forall\ n > 0$

$$\begin{aligned}
\log(n!) &= \log 1 + \log 2 + \cdots + \log(n-1) + \log n \\
&\geq \log\left(\tfrac{n}{2}\right) + \log\left(\tfrac{n}{2} + 1\right) + \cdots + \log(n) \\
&\geq \tfrac{n}{2} \log\left(\tfrac{n}{2}\right)
\end{aligned}$$

$\because \log(n!) = \Omega(n \log(n))$

$\Rightarrow \log(n!) = \Theta(n \log n)$

Problem 3.

```
def binary_representation(n):
    if n > 1:
        binary_representation(n // 2)
    print(n % 2, end='')
```

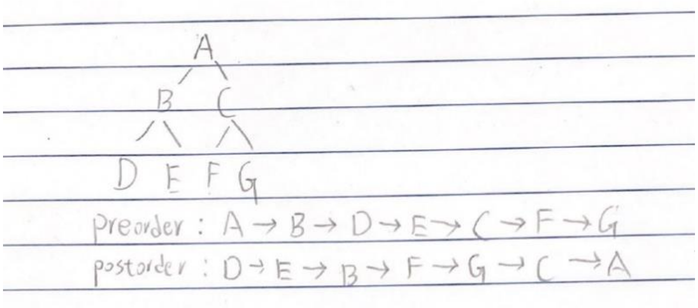
number = int(input("your binary representation is: "))

binary_representation(number)

Problem 4.

Preorder: parent node ,left,right

Postorder: left, right, parent node



- If either the preorder or postorder sequence is empty, return **None**
- The first node in the preorder sequence is the root of the current subtree.
- Create a node with this value.
- Identify the index of the root node in the postorder sequence. This index divides the postorder sequence into left and right subtrees.
- Recursively call the algorithm for the left and right subtrees using the appropriate portions of the preorder and postorder sequences.
- Return the root node.

Example:

```
function reconstruct(preorder, postorder):
    if not preorder or not postorder:
        return None
    root = Node(preorder[0])
    if len(preorder) > 1:
        root.left = reconstruct(preorder[1:1+postorder.index(preorder[1])+1],
postorder[:postorder.index(preorder[1])+1])
        root.right = reconstruct(preorder[postorder.index(preorder[1])+2:],
postorder[postorder.index(preorder[1])+1:-1])
    return root
```