CS5800: Algorithms SEC 05 Summer Full 2024 (Seattle)
Homework 1

Payal Chavan

05/15/2024

Question 1: In each of the following situations, indicate whether $f = \mathcal{O}(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$).

Part (a): $f(n) = n - 100 \qquad g(n) = n - 200$

Solution: Since n-100 and n-200 have the same power of n, we can say that $f = \Theta(g)$.

Part (b): $f(n) = n^{1/2} \qquad g(n) = n^{2/3}$

Solution: If we compare the powers of n i.e., 1/2 and 2/3, we have 0.5 < 0.6. Hence, we can say that $f = \mathcal{O}(g)$.

Part (c): $f(n) = 100n + \log n \qquad g(n) = n + (\log n)^2$

Solution: As both the functions have $\log n$ terms, both are $\mathcal{O}(n)$. Hence, we have $f = \Theta(g)$.

Part (d): $f(n) = n \log n \qquad g(n) = 10n \log 10n$

Solution: After solving $g(n) = 10n \log 10n$, we get $10n(\log 10 + \log n) = 10n \log 10 + 10n \log n$.
Both of them have $f = \mathcal{O}(n \log n)$. Hence, $f = \Theta(g)$.

Part (e): $f(n) = \log 2n \qquad g(n) = \log 3n$

Solution: When we solve $\log 2n = \log 2 + \log n$ and $\log 3n = \log 3 + \log n$, we get $\mathcal{O}(\log n)$. Because, $\log 2$ and $\log 3$ are constant terms. Therefore, $f = \Theta(g)$.

Part (f): $f(n) = 10 \log n \qquad g(n) = \log(n^2)$

Solution: We have, $g(n) = \log(n^2)$ i.e., $2 \log n$. As both of them have $\mathcal{O}(\log n)$, we get $f = \Theta(g)$.

Part (g): $f(n) = n^{1.01} \qquad g(n) = n \log^2 n$

Solution: If we divide, both sides by n, we get $n^{0.01}$ and $\log^2 n$. After comparing these values, ultimately, power function would always overtake the log function. Hence, we get $f = \Omega(g)$.

Part (h): $f(n) = n^2 / \log n \qquad g(n) = n\ (\log n)^2$

Solution: We can neglect $\log n$ part in both the functions, as logs are not that significant. So, then we are left with n^2 and n. Hence, we get $f = \Omega(g)$.

Part (i): $f(n) = n^{0.1} \qquad g(n) = (\log n)^{10}$

Solution: As power function would always be dominant over the log. Hence, we get $f = \Omega(g)$.

Part (j): $f(n) = (\log n)^{\log n} \qquad g(n) = n/\log n$

Solution: Let n>1, then for example $(\log 8)^{\log 8} > 8/\log 8$ i.e., 27 > 2.6
So any number when divided would result in a smaller value as compared to the exponential. Hence, we get $f = \Omega(g)$.

Part (k): $f(n) = \sqrt{n} \qquad g(n) = (\log n)^3$

Solution: Since any polynomial dominates any logarithms, we get $f = \Omega(g)$.

Part (l): $f(n) = n^{1/2} \qquad g(n) = 5^{\log_2(n)}$

Solution: The function $g(n) = 5^{\log_2(n)} = n^{\log_2(5)} = n^{2.32}$.
After comparing, we have $n^{0.5} < n^{2.32}$.
Hence, we get $f = \mathcal{O}(g)$.

Part (m): $f(n) = n2^n \qquad g(n) = 3^n$

Solution: Let us assume, n=k(some real integer). Now, we can neglect the multiplicative constant and so $f(n) = 2^n$.
We will just compare 2^n with 3^n.
By the mathematical induction, we get the result $3^n > 2^n$ for all n and $(n+1)$ values. Therefore, $f = \mathcal{O}(g)$.

Part (n): $f(n) = 2^n \qquad g(n) = 2^{n+1}$

Solution: Since, both differ just by the multiplicative constant 2, we get $f = \Theta(g)$.

Part (o): $f(n) = n! \qquad g(n) = 2^n$

Solution: $n!$ grows very fast than compared to a power of n. It's because the greater number is multiplied with the product each time.
$(n+1)! = 1.2....n.(n+1)$
Whereas, in the case of exponential function
$c^{n+1} = c.c.c.....$
Hence, we get $f = \Omega(g)$.

Part (p): $f(n) = (\log n)^{\log n} \qquad g(n) = 2^{(\log_2 n)^2}$

Solution: Solving, $f(n) = (\log n)^{\log n} = n^{\log(\log n)}$ and
$g(n) = 2^{(\log_2 n)^2} = 2^{(\log_2 n)*(\log_2 n)}$
$g(n) = 2^{(\log_2 n)^{(\log_2 n)}}$
By using Logarithmic rule, $b^{\log_b(k)}$ = k, we get,
$g(n) = n^{(\log_2 n)}$
Now comparing f(n) and g(n),
f(n) < g(n), because $\log(\log n)$ is much slower than that of $\log_2(n)$.
Therefore, we get, $f = \mathcal{O}(g)$.

Part (q): $f(n) = \sum_{i=1}^{n} i^k \qquad g(n) = n^{k+1}$

Solution: Simplifying, $f(n) = \sum_{i=1}^{n} i^k = 1^k + 2^k + 3^k + \cdots + n^K$
$g(n) = n^{k+1} = n^k.n$
Let us assume, $k = 1$(some real integer) and $n = 10$, we get,
$f(n) = 1 + 2 + 3 + \cdots + 10 = 55$
$g(n) = 10^2 = 10 * 10 = 100$
Therefore, as $\sum_{i=1}^{n} i^k < n^{k+1}$, we get $f = \mathcal{O}(g)$.

Question 2: Show that, if c is a positive real number, then $g(n) = 1 + c + c^2 + \cdots + c^n$ is:

(a) $\Theta(1)$ if $c < 1$

(b) $\Theta(n)$ if $c = 1$

(c) $\Theta(c^n)$ if $c > 1$

Solution:
a) if c < 1

Proof. given: $g(n) = 1 + c + c^2 + \cdots + c^n$
Let us consider, c = 0.5
Now, $g(n) = 1 + (0.5) + (0.5)^2 + \cdots + (0.5)^n$
The geometric series goes on decreasing. So, we could skip the decimal terms as they are negligible. The maximum number will be 1. With that, g(n) becomes independent of n.
Hence, we get, $g(n) = \Theta(1)$ □

b) if c = 1

Proof. given: $g(n) = 1 + c + c^2 + \cdots + c^n$
Let us consider, c = 1
Now, $g(n) = 1 + (1) + (1)^2 + \cdots + (1)^n$
The geometric series is unchanging and will result in 'n' terms.
Hence, we get, $g(n) = \Theta(n)$ □

c) if c > 1

Proof. given: $g(n) = 1 + c + c^2 + \cdots + c^n$
Let us consider, c = 2
Now, $g(n) = 1 + (2) + (2)^2 + \cdots + (2)^n$
The geometric series is increasing and the largest term would be $(2)^n$.
Hence, we get, $g(n) = \Theta(c^n)$ □

Question 3: Is there a faster way to compute the nth Fibonacci number than by fib2? One idea involves matrices. We start by writing the equations $F_1 = F_1$ and $F_2 = F_0 + F_1$ in matrix notation:

$$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

Similarly,

$$\begin{pmatrix} F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^2 \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

and in general

$$\begin{pmatrix} F_n \\ F_n + 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

So, in order to compute F_n, it suffices to raise this $2 * 2$ matrix, call it X, to the nth power.

Part (a): Show that two 2x2 matrices can be multiplied using 4 additions and 8 multiplications.

Solution: Let us consider the matrices X and Y.

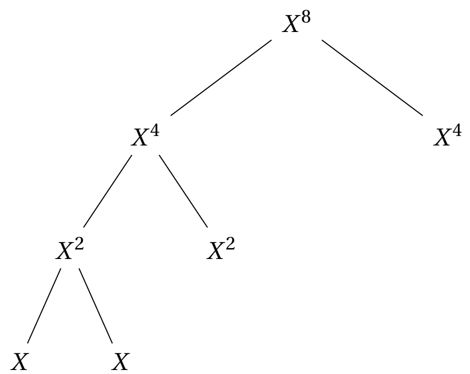
X = $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $and\ Y = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$
Now, multiplying X and Y, we get,

X $\cdot Y = \begin{bmatrix} (1*5)+(2*7) & (1*6)+(2*8) \\ (3*5)+(4*7) & (3*6)+(4*8) \end{bmatrix}$

Hence, from the above 2x2 matrix, we have got 4 additions and 8 multiplications.

Part (b): But how many matrix multiplications does it take to compute X^n? Show that $f = \mathcal{O}(\log n)$ matrix multiplications suffice for computing X^n.(Compute X^8).

Solution: Suppose, we want to compute X^8.



From the above figure, we can write X^8 as,
$X^8 = X^4 * X^4$
$\Longrightarrow (X^2 * X^2) * X^4$
$\Longrightarrow (X * X) * X^2 * X^4$

As we can see, this gives us a total of 3 matrix multiplications.
Then, $X^8 = 3$ multiplications.

To prove, $f = \mathcal{O}(\log n)$,

Here $n = 8$
Taking log on both sides,
$\log_2(n) \Longrightarrow \log_2(8)$
$\Longrightarrow \log_2(2)^3 = 3$

Hence proved $f = \mathcal{O}(\log n)$ for X^n.

If we have to compute Fibonacci of, say 7.
We already know the value of matrix X^8.
All we have to do is compute $X^2 = X * X$ and $X^4 = X^2 * X^2$ and $X^8 = X^4 * X^4$, which gives us a total of 3 matrix multiplication operations as before.
for Even $X^8 \Longrightarrow (X * X) * X^2 * X^4$
for Odd $X^7 = X^{8-1}$

$\implies X^1 * X^2 * X^4$
$\implies X^1 * (X * X) * X^4$

So, as proven before for n = even, similarly for (n-1) is odd, we get same number of matrix multiplications which is O($\log n$).

To generalize for even and odd cases.
$2^n + 1$ to 2^{n+1} exponent ranges where we will have same number of matrix multiplications.
Let's assume $n = 3$, $2^3 + 1$ to 2^{3+1} $\implies$X^9 to X^{16} we will have same number of matrix multiplications that is 4.

Reflection:
From exercise 0.1, I gained the understanding of "Asymptotic Analysis" and the difference between Big $\mathcal{O}$ vs Big Theta Θ vs Big Omega Ω.
From exercise 0.2, I grasped the concept of geometric series.
And finally from exercise 0.4, I understood the working of Fibonacci series and how to compute matrices using the recursive method.

Acknowledgements:
I would like to express my sincere gratitude to the following individuals and resources for their invaluable contributions to this assignment:
1) Prof. Bruce Maxwell: Thank you, Professor Bruce Maxwell for your guidance in this assignment. Your expertise in algorithms greatly influenced my work.
2) Classmates and TA's: I appreciate the discussions of my classmates, and TA's who clarified my doubts.
3) DPV Algorithms Textbook: This textbook was a useful resource for me to understand the basics of algorithms.
4) https://www.geeksforgeeks.org/difference-between-big-oh-big-omega-and-big-theta/: This website provided clear explanations of various notations, enhancing my understanding.
5) https://www.mathsisfun.com/algebra/matrix-multiplying.html: This website helped me to learn matrix multiplication.
6) https://www.chilimath.com/lessons/advanced-algebra/logarithm-rules/: This website provided me the logarithmic rules to solve exercise 0.1 questions.