Applied Algorithms

Written Assignment-1

Jaya Sandeep Ketha

September 2023

Q1. Suppose $\mathbf{f(x) = 10x^2 + 5x + 3}$ and $\mathbf{g(x) = x^3 + x - 100}$. Recall the formal definitions of big-Oh. Show that $\mathbf{f(x) = O(g(x))}$ using the definition of asymptotic complexity that we saw; don't forget to show your constants and how they work to prove what you want.

Answer:
Definition of Big-Oh:
The function $f(n) = O(g(n))$ iff $\exists$ positive constants C and n_0 such that $f(n) \leq Cg(n)$ for all $n \geq n_0$.

Given functions:

$$f(x) = 10x^2 + 5x + 3$$
$$g(x) = x^3 + x - 100$$

Required to show that $f(x) = O(g(x))$.
By the definition of Big O, $f(x) = O(g(x))$ if there exist positive constants C and x_0 such that

$$10x^2 + 5x + 3 \leq C(x^3 + x - 100) \text{ for all } x \geq x_0.$$

By definition, we know that $C > 0$ and $x_0 > 0$, which implies $x > 0$.

Since $x > 0$, we can see that $10x^2 + 5x + 3$ is always positive. Therefore, $x^3 + x - 100$ is also positive, as C is also positive.
Thus, we have:

$$x^3 + x - 100 > 0$$
$$x^3 + x > 100$$

This inequality is valid only when $x \geq 5$.
From this, it is clear that $x_0 = 5$. Now, with $x_0 = 5$,

$$10(5^2) + 5(5) + 3 \leq C(5^3 + 5 - 100)$$

Simplifying:

$$278 \leq 30C$$
$$C \geq 9.266$$

From this, it is clear that $C = 10$ and $x_0 = 5$, makes the given inequality valid.
So, there exist positive constants C and x_0 for which $f(x) \leq C(g(x))$ for all $x \geq x_0$.
Hence, we can say that $f(x) = O(g(x))$.

Validation:

Let us consider $x = 4$, 5, and 6 to validate our results.
@ $x = 4$:

$$10(4^2) + 5(4) + 3 \leq C(4^3 + 4 - 100)$$

Simplifying:

$$183 \leq -30C$$

$$C \leq -\frac{183}{30}$$

This shows that C must be negative, which contradicts the definition of Big O. So $x_0 \neq 4$.

@ $x = 5$:

$$10(5^2) + 5(5) + 3 \leq 10(5^3 + 5 - 100)$$

Simplifying:

$$289 \leq 300$$

This satisfies the definition of Big O.
@ $x = 6$:

$$10(6^2) + 5(6) + 3 \leq 10(6^3 + 6 - 100)$$

Simplifying:

$$393 \leq 1220$$

This satisfies the definition of Big O.
Hence, there exists a positive C and x_0 such that $f(x) \leq C \cdot g(x)$ for all $x \geq x_0$.
So, f(x) = O(g(x)).

Q2. Consider the following: $\mathbf{i, j, k, l, n}$ are integers. What is the exact value of f(n), which is returned by the algorithm (i.e., the final value of k, in terms of n? What is the asymptotic complexity of this value in terms of n in the big-Oh (or big-Theta if you prefer that) notation? What is the asymptotic complexity of the running time of this algorithm, again in the big-Oh (or big-Theta) notation? Show your analysis to get the simplest but tightest results wherever we're asking for asymptotic complexity. For instance, we would like to see $\mathcal{O}(n^2)$ over $\mathcal{O}(2n^2+n)$.

```
f(n):
    k = 0; l = 1;
    for i = 1 to n do
        for j = 1 to i do
            k = k + 1
    for i = 1 to n do
        k = k + 1
        l = l * 2
    return k
```

Answer:
Initialization:

k is initialized to 0.

l is initialized to 1.

First Nested Loop:

The outer loop runs from $i = 1$ to n.

The inner loop runs from $j = 1$ to i.

In each iteration of the inner loop, k is incremented by 1.

Therefore, the inner loop contributes $\left(\frac{n \cdot (n+1)}{2}\right)$ to the value of k.

This is an arithmetic progression with a sum of $\frac{n \cdot (n+1)}{2}$.

Second Loop:

The loop runs from $i = 1$ to n.
In each iteration of this loop, k is incremented by l, and then l is doubled.
Initially, l is 1, and it doubles in each iteration.
So, k is incremented by $1, 2, 4, 8, \ldots$ in the first, second, third, fourth, ... iterations, respectively.

Return Value: The final value of k is returned.

Exact value of f(n):

The final value of k is the sum of the contributions from the two loops:

$$k = \frac{n \cdot (n+1)}{2} + 1 + 2 + 4 + 8 + \ldots + 2^{n-1}$$

This can be simplified to:

$$k = \frac{n \cdot (n+1)}{2} + 2^n - 1$$

Asymptotic Complexity of f(n):

The dominant term in the above expression is 2^n.
Therefore, the asymptotic complexity of $f(n)$ is $O(2^n)$.

Asymptotic Complexity of Running Time:

The running time of the algorithm is determined by the number of iterations in the loops.

The first nested loop runs in $O(n^2)$ time because the sum of the first n natural numbers is $\dfrac{n \cdot (n+1)}{2}$.

The second loop runs in $O(n)$ time because it iterates n times, and each iteration takes constant time.
Combining both loops, the overall running time is
$O(n^2) + O(n) = O(n^2)$ because the nested loop dominates the complexity.

The exact value of $f(n)$ is $\frac{n \cdot (n+1)}{2} + 2^n - 1$.

The asymptotic complexity of $f(n)$ is $O(2^n)$.

The asymptotic complexity of the running time is $O(n^2)$.

Q3. Show that $\frac{1}{n^2} \in O\left(\frac{1}{n}\right)$ showing the constants, etc.

Answer:
By definition, $f(n) = O(g(n))$ if there exist positive constants C and n_0 such that $f(n) \leq C \cdot (g(n))$ for all $n \geq n_0$.

Let $f(x) = \frac{1}{n^2}$ and $g(x) = \frac{1}{n}$.

We know that $C > 0$ and $n_0 > 0$, which implies $n > 0$.
Resolving the inequality:

$$\frac{1}{n^2} \leq C \cdot \frac{1}{n}$$

Multiplying both sides by n^2:

$$1 \leq Cn$$

Let us assume $n_0 = 1$, and we need to prove that there exists a positive value of C such that $f(n) \leq C(g(n))$ for all $n \geq 1$.
Upon substituting $n = 1$ into the inequality, we get:

$$C \cdot 1 \geq 1$$

So, $C \geq 1$.

Hence, it is proved that there exists a positive value of C (in this case, $C \geq 1$) such that $1/n^2 \leq C(1/n)$.

Further, even the graph of $\frac{1}{n^2}$ vs $\frac{1}{n}$ shows that $\frac{1}{n^2} \leq C \cdot \frac{1}{n}$ for $C \geq 1$ and $n \geq 1$.

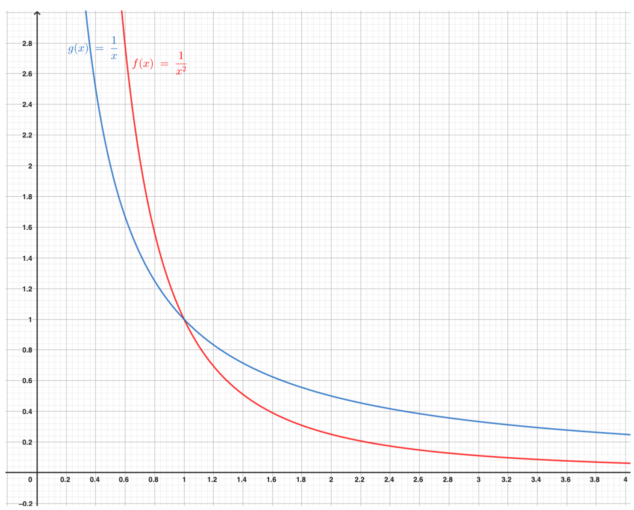

Figure 1: $1/n^2$ vs $1/n$

Thus, $f(n) = O(g(n))$.

Q4. You are given $\mathbf{f(n) = O(g(n))}$ and $\mathbf{f(n) = O(h(n))}$, Give an example where $\mathbf{g(n) = O(h(n))}$ and where $\mathbf{h(n) = O(g(n))}$.

Answer:
Let's assume:

$$f(n) = 3n^2$$
$$g(n) = 5n^3 + n - 2$$
$$h(n) = 7n^3 - n^2 - 2$$

From these equations, we have:

1. $f(n) \leq C(g(n))$:
   $3n^2 \leq C(5n^3 + n - 2)$
   If we choose $n_0 = 1$, then for $n \geq 1$, there exists a positive constant C(true for all $C \geq \frac{3}{4}$)
   such that $f(n) \leq C(g(n))$ for all $n \geq 1$.

2. $f(n) \leq C(h(n))$:
   $3n^2 \leq C(7n^3 - n^2 - 2)$
   If we choose $n_0 = 1$, then for $n \geq 1$, there exists a positive constant C(true for all $C \geq \frac{3}{4}$)
   such that $f(n) \leq C(g(n))$ for all $n \geq 1$.

Therefore, these equations satisfy the given conditions $f(n) = O(g(n))$ and $f(n) = O(h(n))$.
Now to show that $g(n) = O(h(n))$ and $h(n) = O(g(n))$:

3. $g(n) \leq C(h(n))$:
   $5n^3 + n - 2 \leq C(7n^3 - n^2 - 2)$
   If we choose $n_0 = 1$, then for $n \geq 1$, there exists a positive constant C (true for all $C \geq 1$)
   such that $g(n) \leq C(h(n))$ for all $n \geq 1$.

4. $h(n) \leq C(g(n))$:
   $7n^3 - n^2 - 2 \leq C(5n^3 + n - 2)$
   If we choose $n_0 = 1$, then for $n \geq 1$, there exists a positive constant C (true for all $C \geq 1$)
   such that $h(n) \leq C(g(n))$ for all $n \geq 1$.

Hence, the given equations satisfy the required conditions, and we can conclude that $f(n) = O(g(n))$ and $f(n) = O(h(n))$, with $g(n) = O(h(n))$ and $h(n) = O(g(n))$.

Q5. Compare the following pairs of functions, and show which one is big-Oh of the other one (prove using the definition):

Answer:

1. $(n^n, n!)$:
We need to determine whether n^n is $O(n!)$. By definition, $f(n)$ is $O(g(n))$ if there exist positive constants C and n_0 such that $f(n) \leq C \cdot g(n)$ for all $n \geq n_0$.
Let's consider $f(n) = n^n$ and $g(n) = n!$. Now, we have:

$$n^n \leq C \cdot n!$$
$$n \cdot n \cdot n \cdot \ldots \cdot n \leq C \cdot n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$$

From the inequality, it is clear that n^n is always greater than $n!$ for any positive value of C and n, because n^n is n multiplied by itself n times, whereas $n!$ is a factorial that multiplies down to 1, which is always smaller.

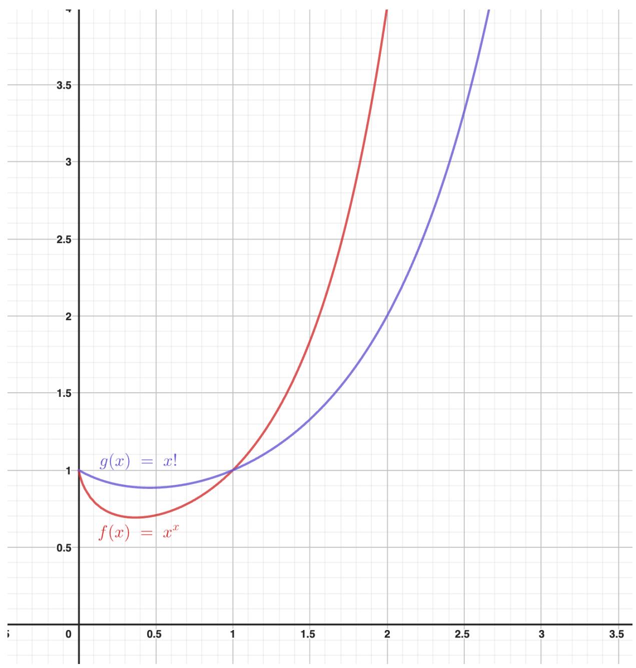

Figure 2: x^x vs $x!$

Furthermore, n^n grows faster than $n!$, so it's not bounded by a constant. Therefore, $n^n \neq O(n!)$.
But $n! = O(n^n)$.

Let $n_o = 1$, and $n! \leq \dot{C}(n^n)$.
We have $n \geq 1$, so for $n = 1$, we have $C \geq 1$. Hence there exists a positive value of C such that $n! \leq C(n^n)$ for all $n \geq 1$. This shows that $n! = O(n^n)$.

2. $(2^n, 3^n)$:

   We need to determine whether 2^n is $O(3^n)$. By definition, $f(n)$ is $O(g(n))$ if there exist positive constants C and n_0 such that $f(n) \leq C \cdot g(n)$ for all $n \geq n_0$.

   Let's consider $f(n) = 2^n$ and $g(n) = 3^n$. Now, we have:

$$2^n \leq C \cdot 3^n$$
$$2 \cdot 2 \ldots 2 \leq C \cdot 3 \cdot 3 \ldots 3$$

   We know that $2 < 3$, and it's clear that for all positive values of $n \geq n_0$ (where n_0 is 0), there exists a positive value $C \geq 1$ such that the inequality $2^n < 3^n$ holds. Therefore, we can conclude that $2^n = O(3^n)$.

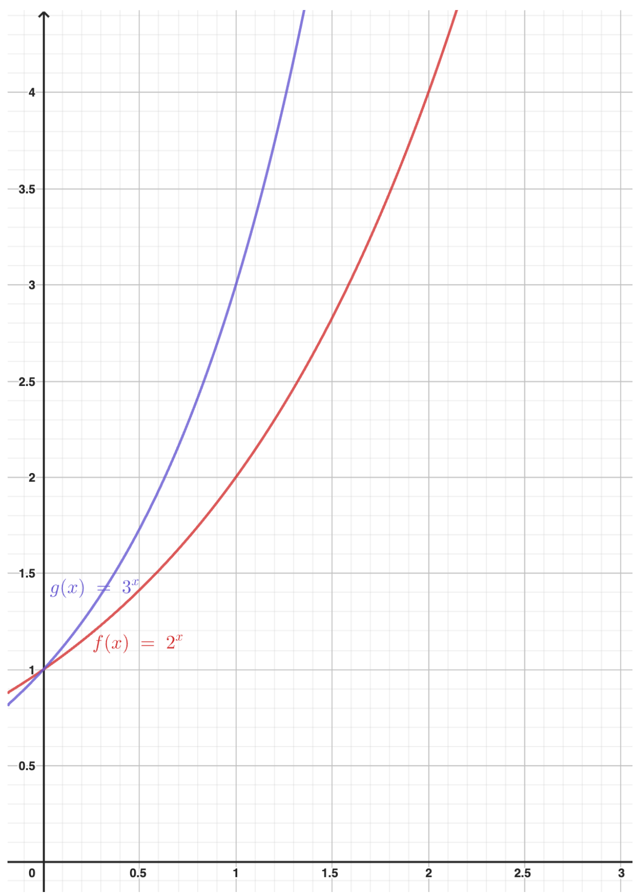

Figure 3: 2^x vs 3^x

3. $(\log n, \log^2 n)$:

   We need to determine whether $\log n$ is $O(\log^2 n)$. By definition, $f(n)$ is $O(g(n))$ if there exist positive constants C and n_0 such that $f(n) \leq Cg(n)$ for all $n \geq n_0$.

   Let's consider $f(n) = \log n$ and $g(n) = \log^2 n$. Now, we have:

$$\log n \leq C \cdot \log^2 n$$

   We know that $\log^k n = (\log n)^k$, which gives us $\log^2 n = (\log n)^2$.

$$\log n \leq C \cdot (\log n)^2$$

If the logarithm is applied on both sides for comparison:

$$\log(\log n) \leq 2\log(\log n)$$

From this, it is clear that $\log(\log n) < 2\log(\log n)$ for all n ≥ 10. Hence, $\log n \leq C{\cdot}(\log^2 n)$ for any value of C ≥ 1.

We also know from below graph that $\log n \leq \log^2 n$ for $n \geq 10$ for all values of C.
Therefore, $\log n = O(\log^2 n)$.

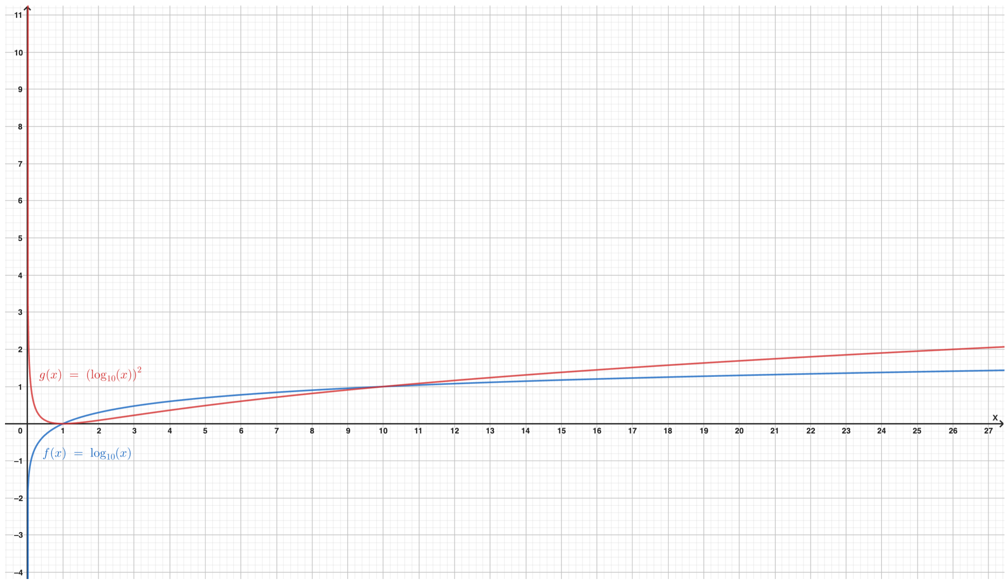

Figure 4: $\log x$ vs $\log^2 x$

4. (n$^{\sqrt{n}}$, n^n):

To prove that n$^{\sqrt{n}} = O(n^n)$, we need to show that there exist positive constants C and n_o such that for all $n \geq n_o$, the following inequality holds:

$$\mathrm{n}^{\sqrt{n}} \leq C \cdot n^n \quad \text{for } n \geq n_o$$

First, let's simplify the inequality by taking the absolute value and dividing both sides by n^n:

$$\frac{n^{\sqrt{n}}}{n^n} \leq C$$

We can rewrite $n^{\sqrt{n}}$ as $n^{n^{1/2}}$ and n^n as n^{n^1} to make it easier to work with:

$$\frac{n^{n^{1/2}}}{n^{n^1}} \leq C$$

Now, apply the properties of exponents. When you divide two numbers with the same base, you subtract the exponents:

$$n^{(n^{1/2}-n^1)} \leq C$$

Now, we need to determine if there is a constant C such that $n^{(n^{1/2}-n^1)} \leq C$ for all sufficiently large n.

To simplify further, let's analyze the exponent $(n^{1/2} - n^1)$:

$$(n^{1/2} - n^1) = n^{1/2} - n$$

Now, we can see that as n becomes larger, $n^{1/2}$ grows much slower than n because the exponent of $n^{1/2}$ is smaller than the exponent of n. Therefore, as n approaches infinity, $(n^{1/2} - n^1)$ approaches negative infinity. So, for sufficiently large n, we can say that $(n^{1/2} - n^1) < 0$.

Now, let's rewrite the inequality with this knowledge:

$n^{(n^{1/2} - n^1)} < 1$ (because any positive number raised to a negative power is less than 1 but greater than 0)

Now, we have:

$$n^{(n^{1/2} - n^1)} < 1 \leq C$$

We can see that for all sufficiently large n, this inequality holds. Therefore, we have shown that $n^{\sqrt{n}} = O(n^n)$ with constants $C = 1$ and n_o (the value of n_o may be a very large number, but it exists). This proves the statement.

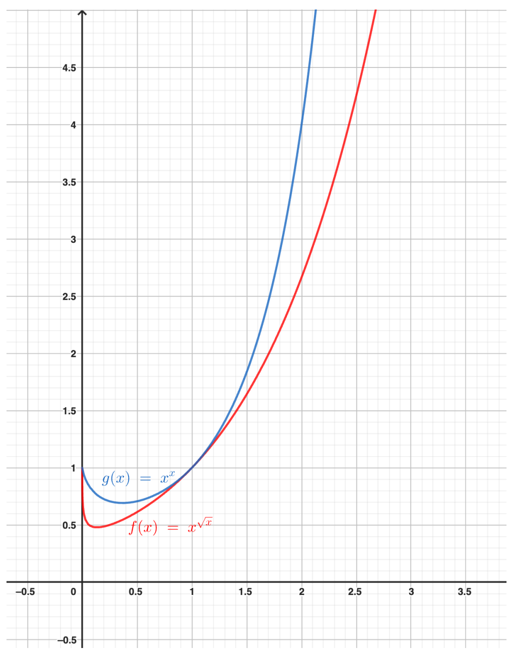

Figure 5: $x^{\sqrt{x}}$ vs x^x

Let us assume $n_0 = 1$. We have $n \geq 1$, so $1^{\sqrt{1}} \leq C{\cdot}1^1$.

Solving for C:

$$C \geq 1$$

Hence, there exists a positive C such that $f(x) \leq C \cdot g(x)$ for all $n \geq 1$. We can say $n^{\sqrt{n}} = O(n^n)$.

Q6. Bonus Question: Given two non-negative valued functions $\mathbf{f(n)}$ and $\mathbf{g(n)}$, is it possible that neither is big-Oh of the other; i.e., can we have both $\mathbf{f(n) \neq O(g(n))}$ and $\mathbf{g(n) \neq O(f(n))}$? Argue.

Answer:
Yes, it is possible to have two functions that are neither Big-Oh of the other. Let us consider the following examples:
Example 1:

$$f(x) = \cos(x) + 1$$
$$g(x) = 1$$

In this example, $f(x)$ is an oscillating function. Its values can be greater than $g(x)$ (where $g(x) = 1$) for some intervals and then drop below $g(x)$ later, with this cycle repeating. This behavior shows that there is no fixed upper boundary, so there are no constants C and n_o such that neither $f(x) \leq C \cdot g(x)$ nor $g(x) \leq C \cdot f(x)$ holds for all $n \geq n_o$. Therefore, for these two functions, $f(x) \neq O(g(x))$ and $g(x) \neq O(f(x))$.

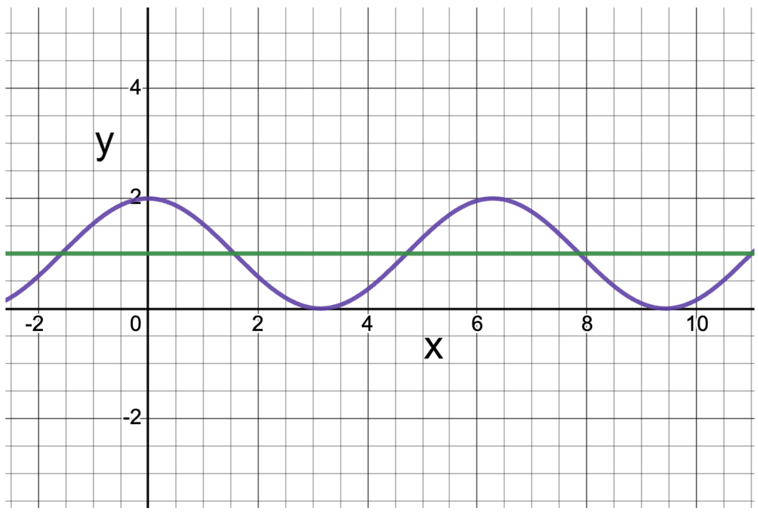

Figure 6: (cos x + 1) vs 1

Example 2:

$$f(x) = n$$
$$g(x) = \begin{cases} n^2 & \text{if } n \text{ is even} \\ 1 & \text{if } n \text{ is odd} \end{cases}$$

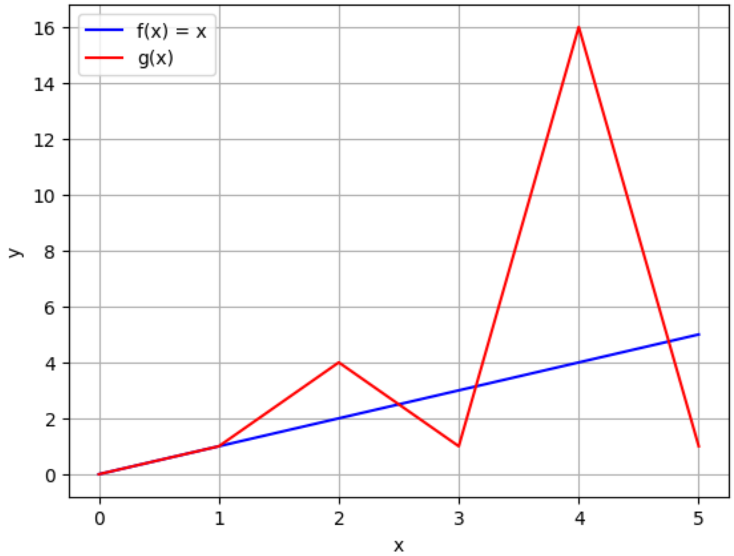

Figure 7: f(x) vs g(x)

In this case, $g(x)$ is a function that varies based on the value of n. There are situations where $g(x)$ can be greater than $f(x)$ for certain positive constants C and n_o, where $n \geq n_o$, but this is not true for all values of n. Similarly, this applies vice-versa too.
Let n is even,

$$\begin{aligned} f(x) = n \text{ and } g(x) = n^2. \\ f(x) \leq C \cdot g(x) \\ n \leq C \cdot n^2 \end{aligned}$$

By choosing $n_o = 1$, we have $n \geq 1$, so we get $C \geq 1$. Hence, there exist constants C and n_o such that $f(x) = O(g(x))$. However, the reverse is not true ($g(x) \leq C \cdot f(x)$), and $g(x) \neq O(f(x))$ because n^2 grows at a faster rate than n (i.e., $n^2 \geq n$ for $n \geq 1$).

Now, let us consider the case where n is odd, and $C = 1$ from the previous equation:

$$f(x) = n$$
$$g(x) = 1$$

Here, $f(x)$ is always greater than or equal to $g(x)$ (i.e., $n \geq 1$), for all $n \geq 1$. While it's opposite for $n < 1$. There is a constant fluctuation in the values which are greater among $f(x)$ and $g(x)$ for both even and odd n.
Hence for these two functions, $f(x) \neq O(g(x))$ and $g(x) \neq O(f(x))$.

Conclusion:
These examples illustrate that it is possible to have functions $f(n)$ and $g(n)$ where neither function is Big-Oh of the other ($f(n) \neq O(g(n))$ and $g(n) \neq O(f(n))$).