Asymptotic Analysis of Expressions

Basic Problems 2: Homework

Mpangi Erick Musungu

April 25, 2025

1 INTRODICTION

This document illustrates the visual comparison of various functions used in asymptotic analysis. The plots support our theoretical growth comparisons using Θ-notation.

This work focuses on analyzing and comparing a series of representative functions commonly used to express algorithmic complexity, including logarithmic, polynomial, exponential, and factorial-based expressions. By plotting these functions over a shared domain, we aim to visualize their respective growth rates and better grasp the asymptotic differences between them.

The goal of this analysis is not only to provide a clear graphical comparison but also to reinforce the importance of selecting efficient algorithms in real-world applications, particularly where input sizes can grow significantly. Through this comparative study, we seek to build deeper intuition about which algorithms are practical and scalable, and which become computationally infeasible as input sizes increase.

E1: Graphical Representation of Expressions

We analyze the following expressions in terms of Θ-order and display their graphical growth. Expressions:

1. $n \log n$
2. n^{-1}
3. $\log n$
4. $n^{\log n}$
5. $10n + n^{3/2}$
6. π^n
7. 2^n
8. $2^{\log n}$
9. $2^{\log^2 n}$
10. $\log n!$

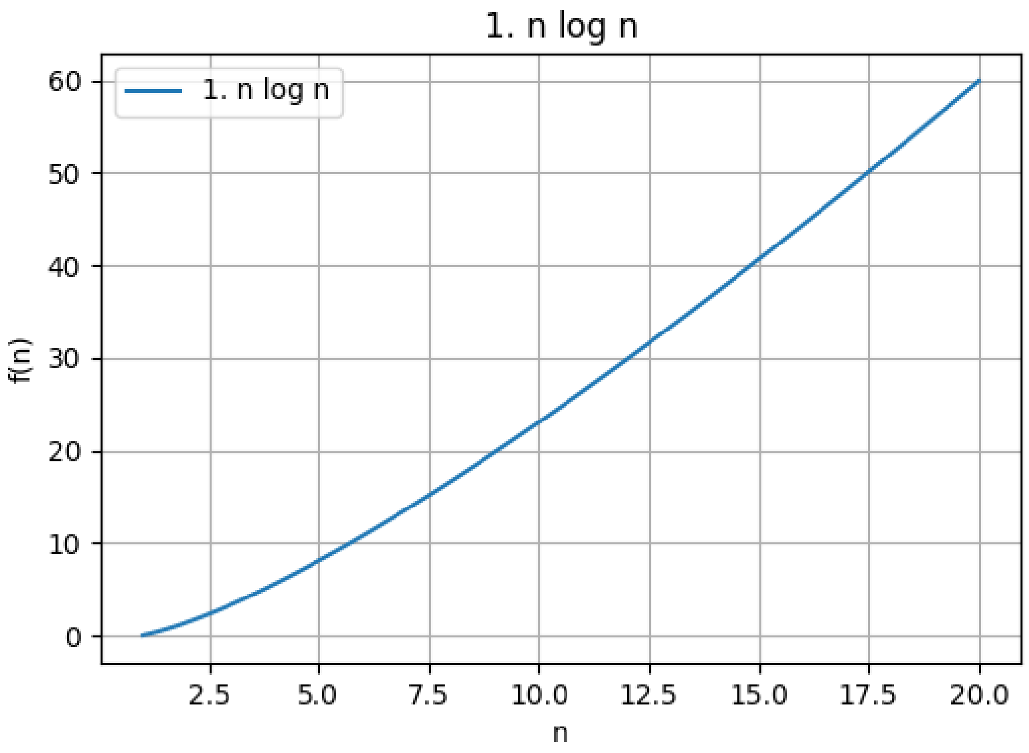

Figure 1: Graphical comparison of n log n.

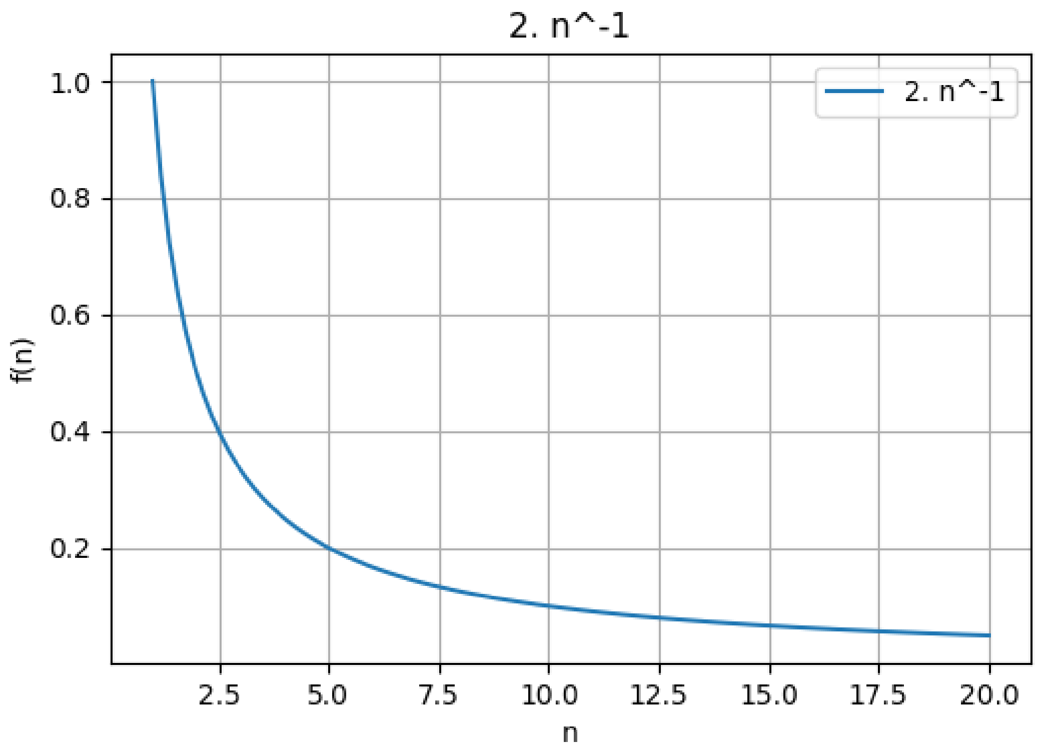

Figure 2: Graphical comparison of n^{-1}.

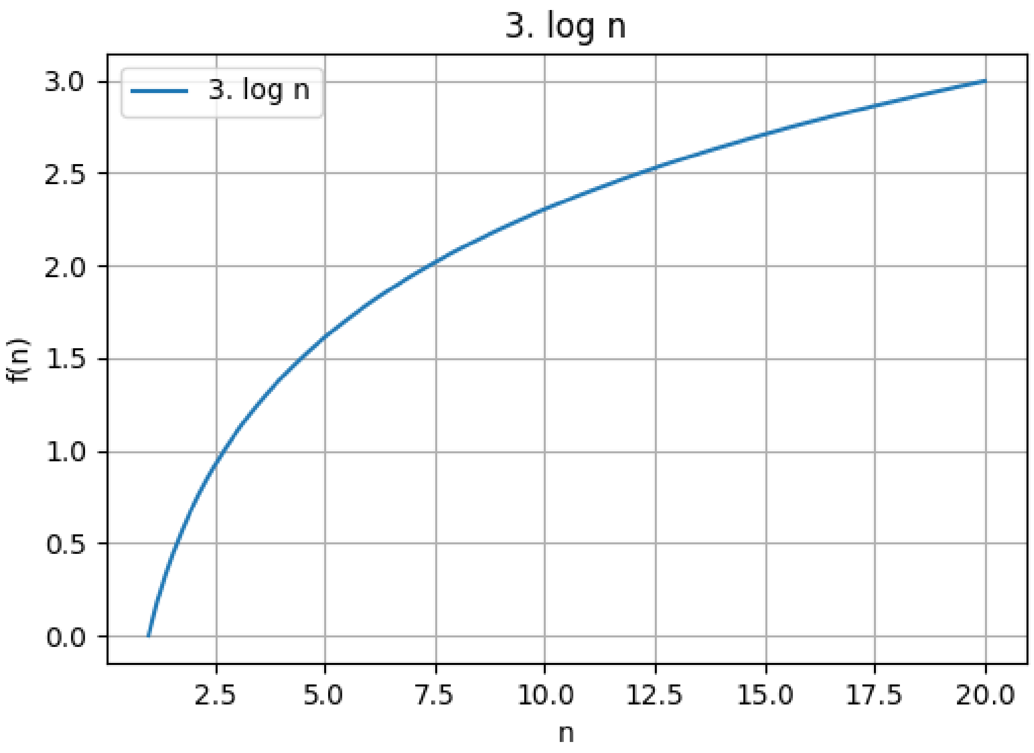

Figure 3: Graphical comparison of log n.

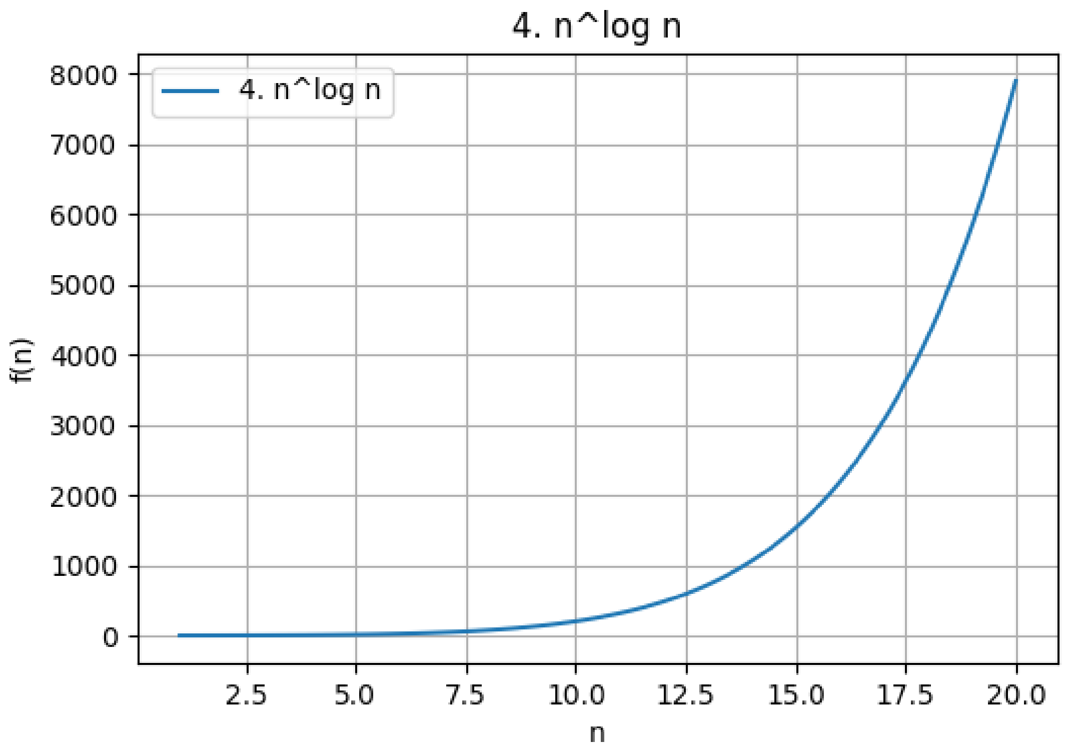

Figure 4: Graphical comparison of n^{logn}.

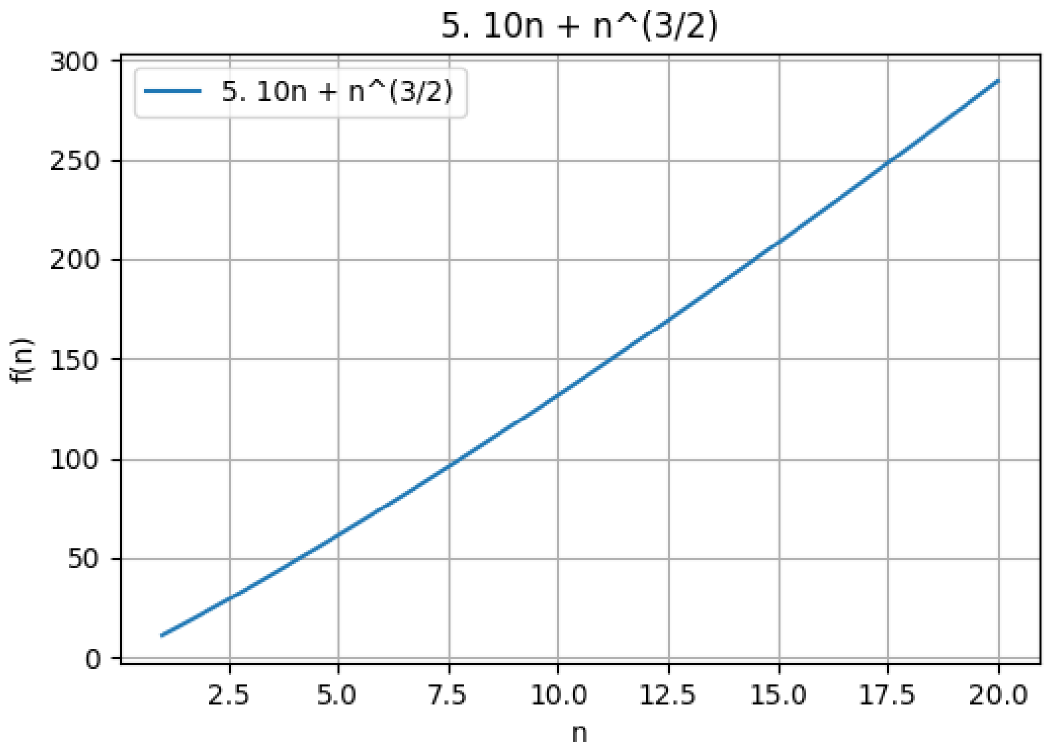

Figure 5: Graphical comparison of $10n + n^{3/2}$.

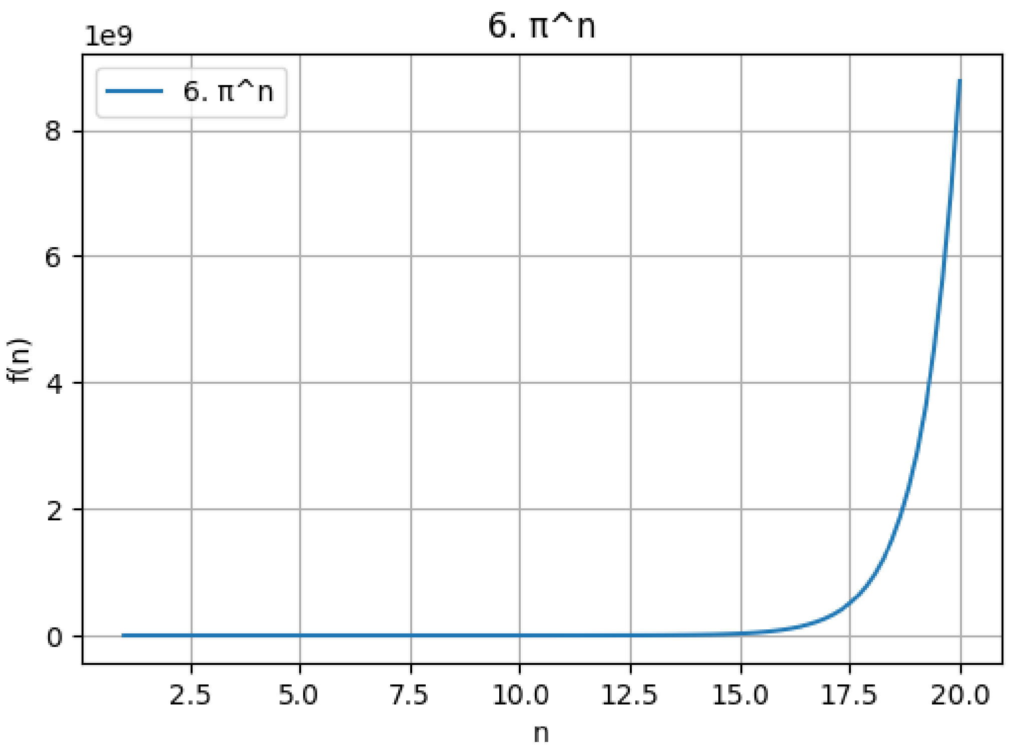

Figure 6: Graphical comparison of π^n .

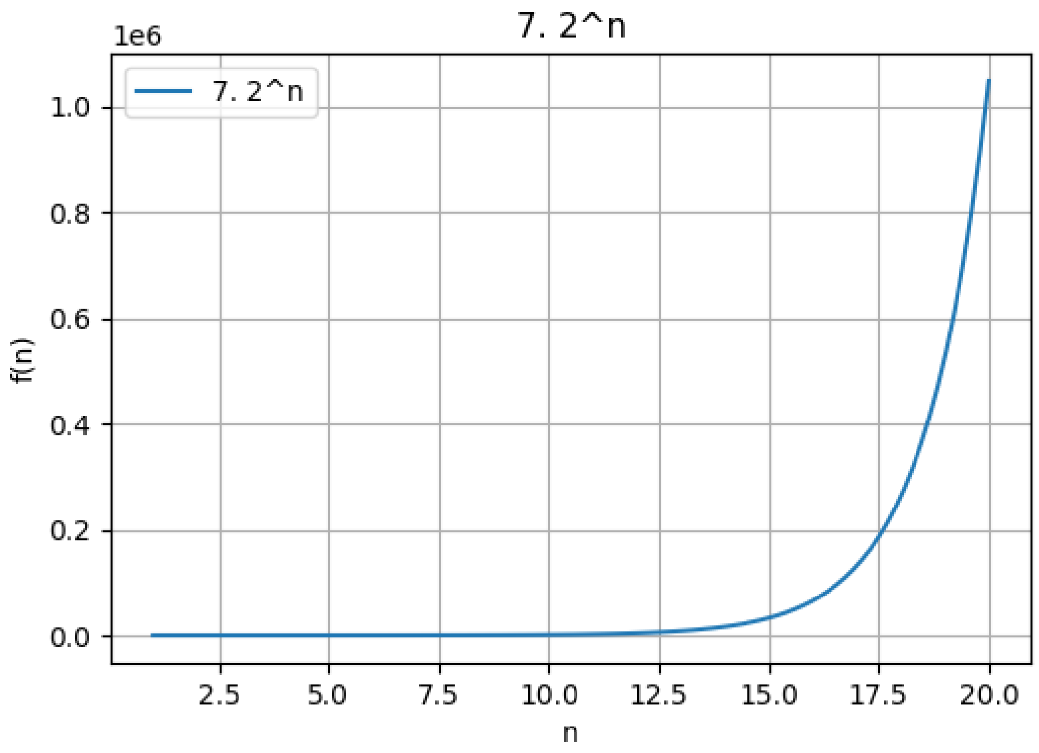

Figure 7: Graphical comparison of 2^n .

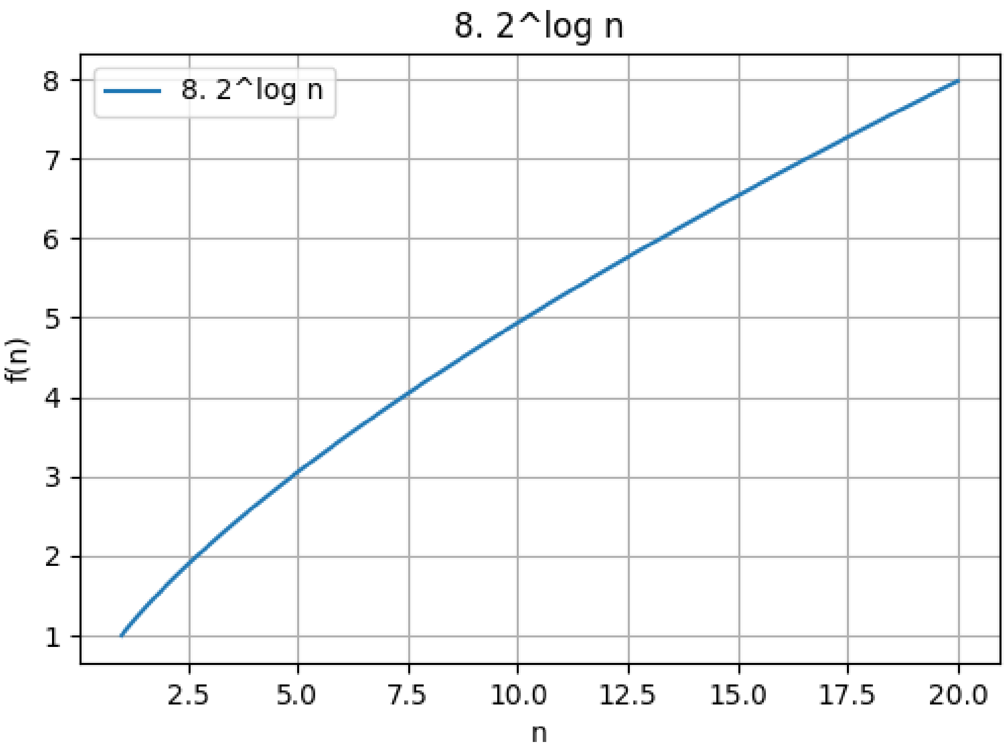

Figure 8: Graphical comparison of 2^{logn} .

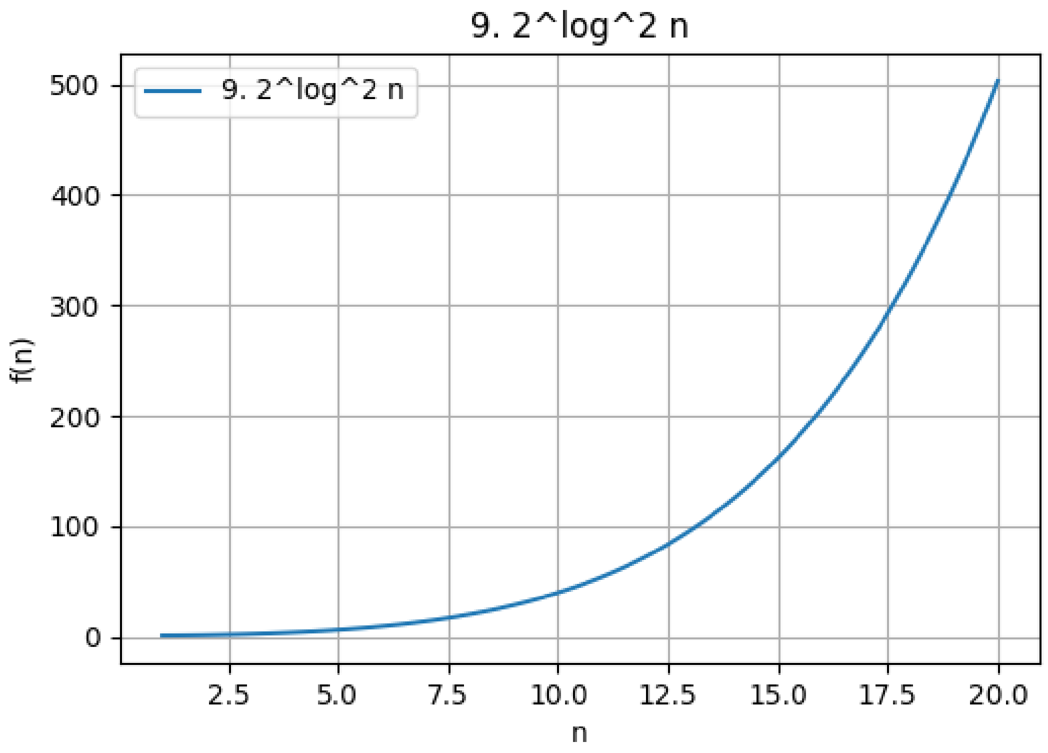

Figure 9: Graphical comparison of $2^{2^{log n}}$.

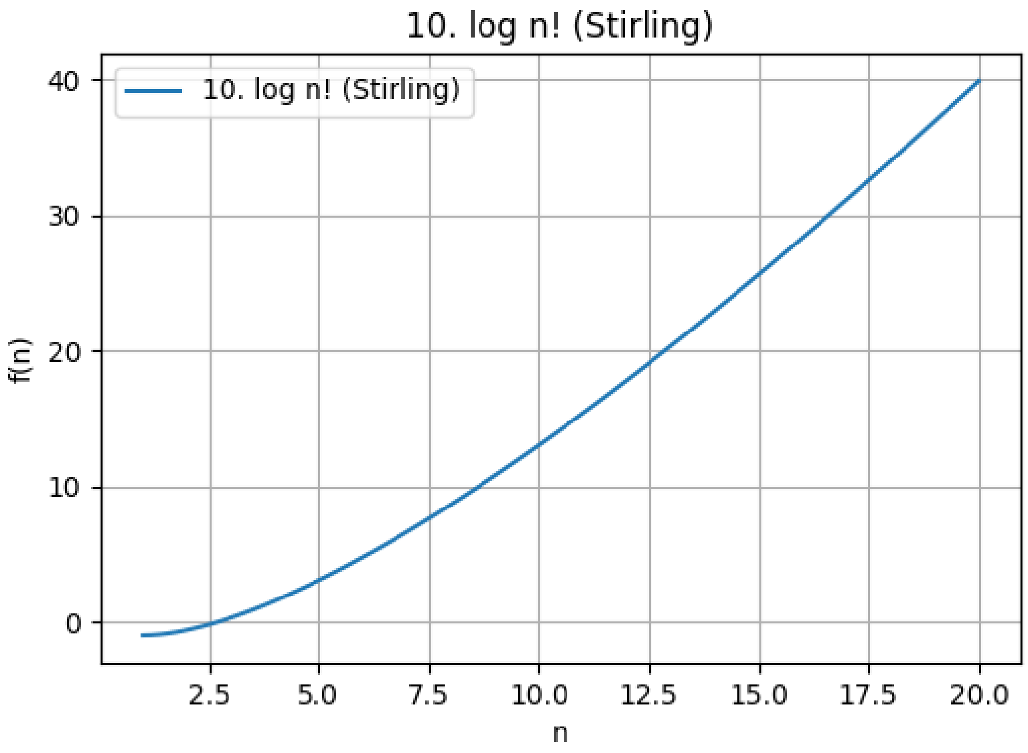

Figure 10: Graphical comparison of $log\ n!$.

Note: $\log n!$ is approximated using Stirling's approximation: $\log n! \approx n \log n - n$.

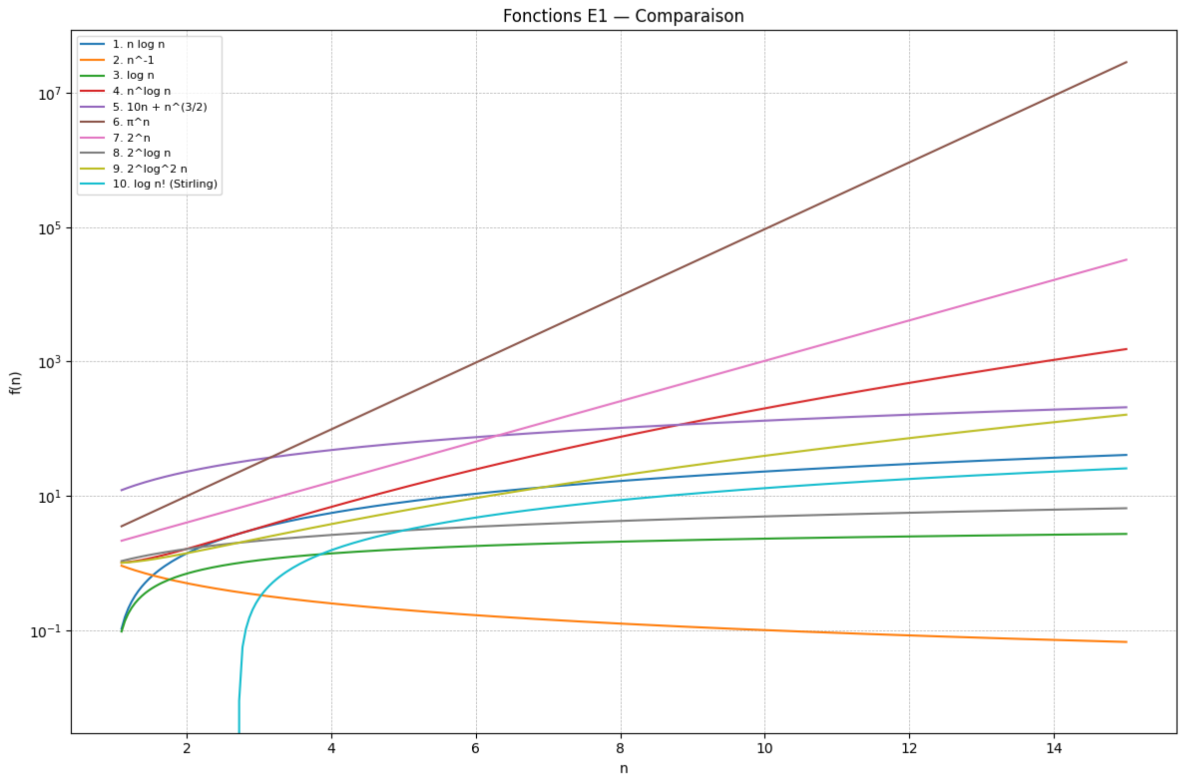

Figure 11: Graphical comparison of E1 expressions.

E2: Ordering Expressions by Θ-Growth

We have:

1. 2^{2^n}
2. 2^{n^2}
3. $n^2 log\ n$
4. n
5. n^{2n}

Increasing Order of Growth

By asymptotic comparison, the increasing order of growth is:

$$n \prec n^{2\log n} \prec 2^n \prec 2^{n^2} \prec n^{2n}$$

Graphical Representation

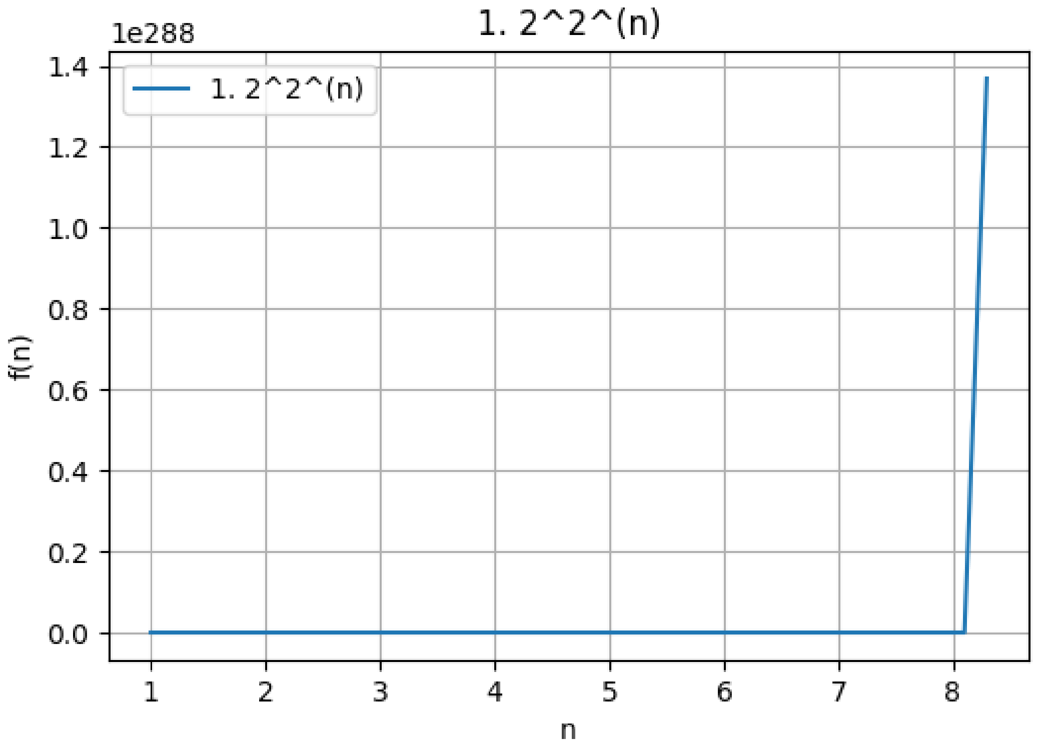

Figure 12: Graphical growth comparison of 2^{2^n}.

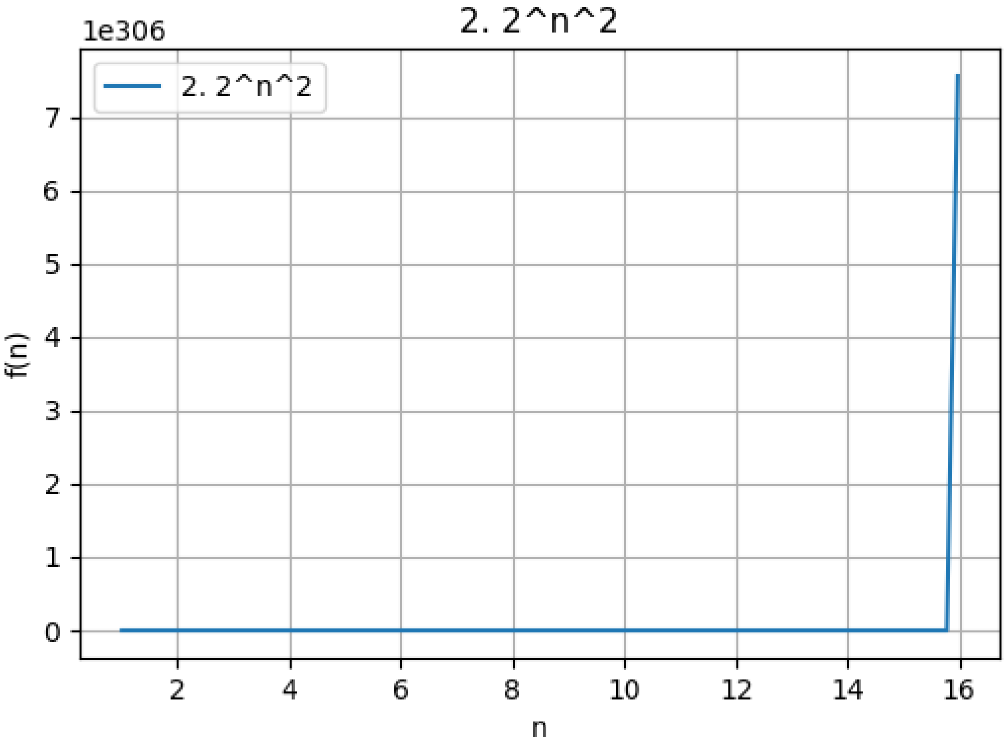

Figure 13: Graphical growth comparison of 2^{n^2}.

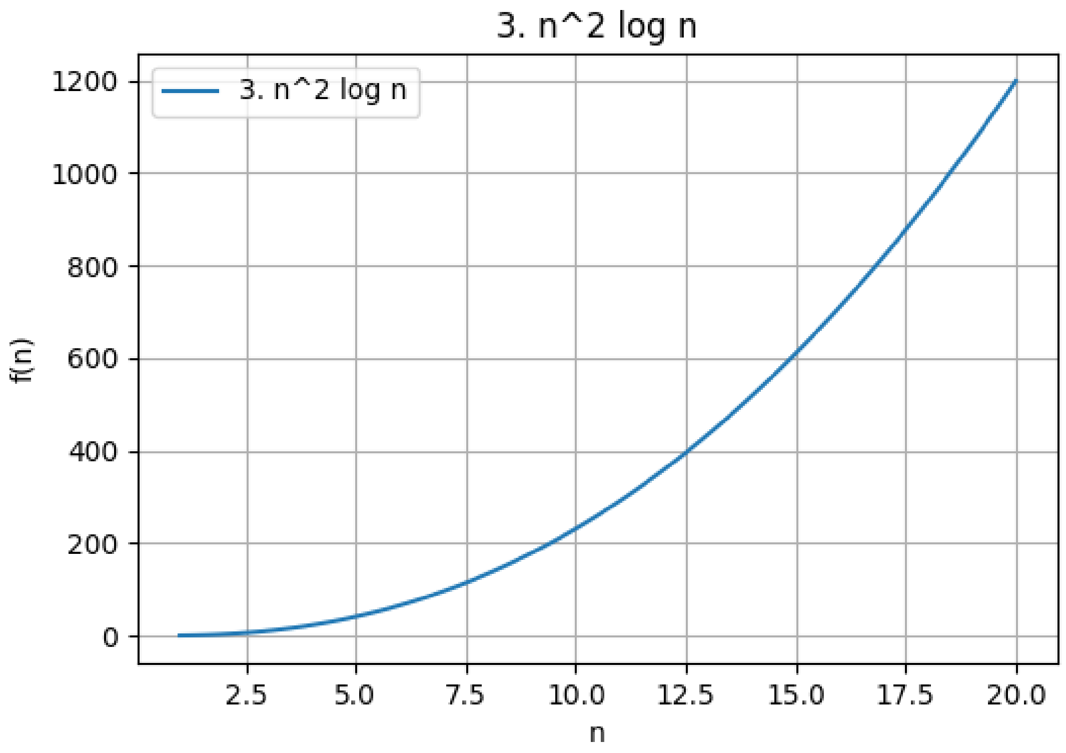

Figure 14: Graphical growth comparison of n^2 *log* n.

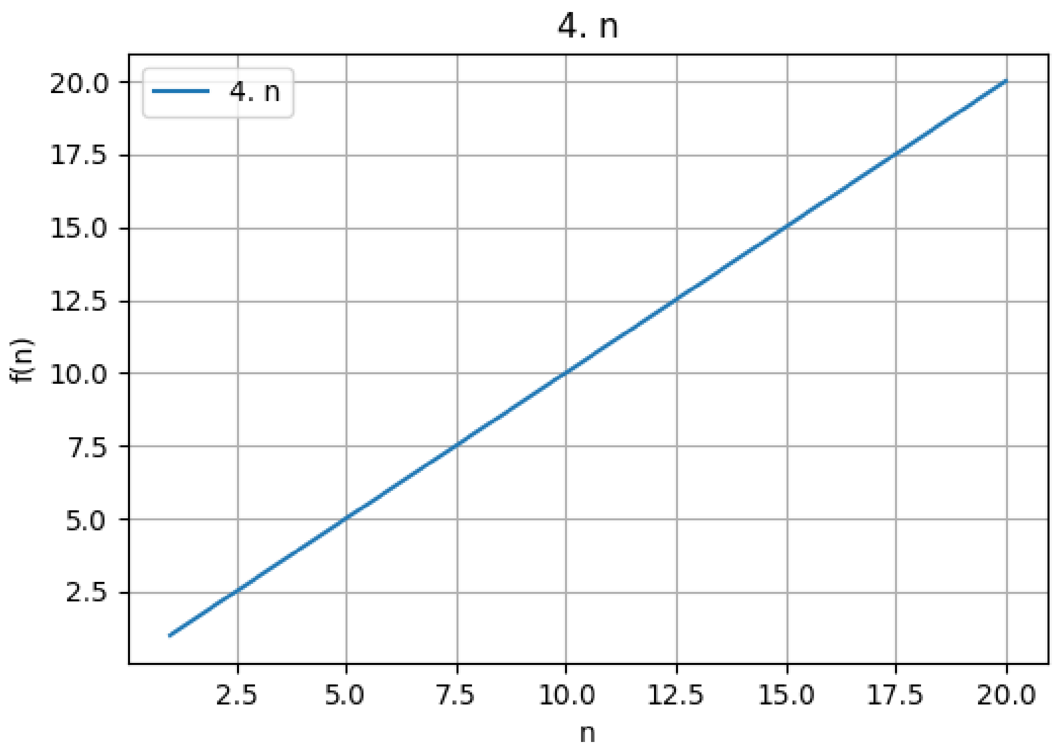

Figure 15: Graphical growth comparison of n.

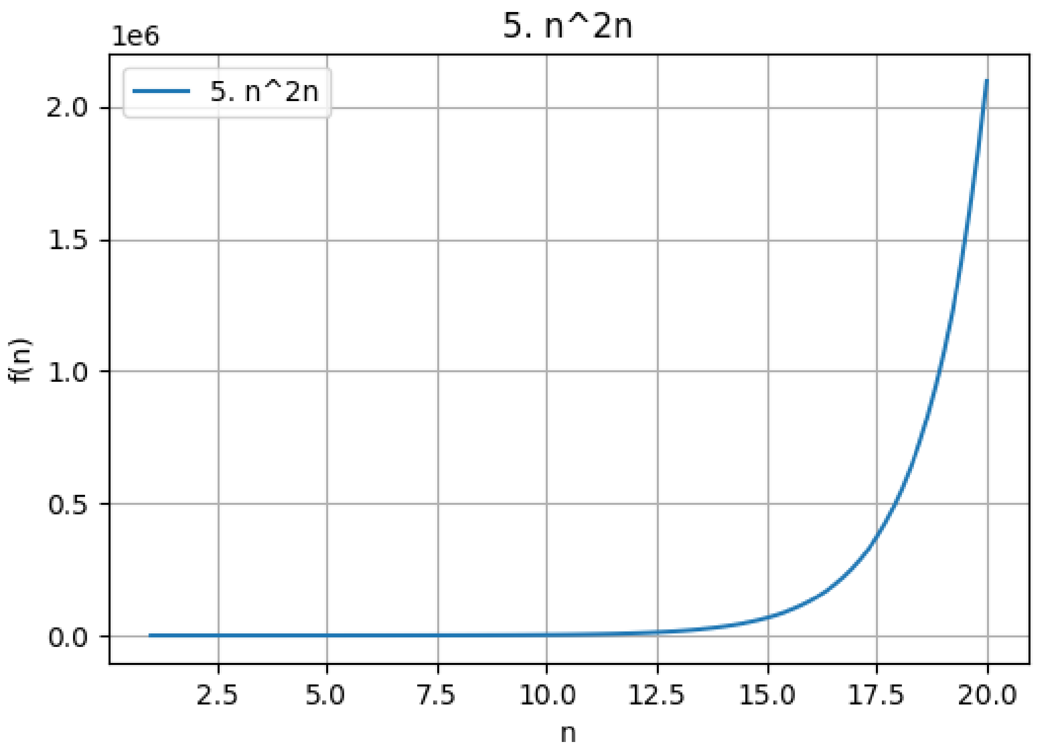

Figure 16: Graphical growth comparison of n^{2n}.

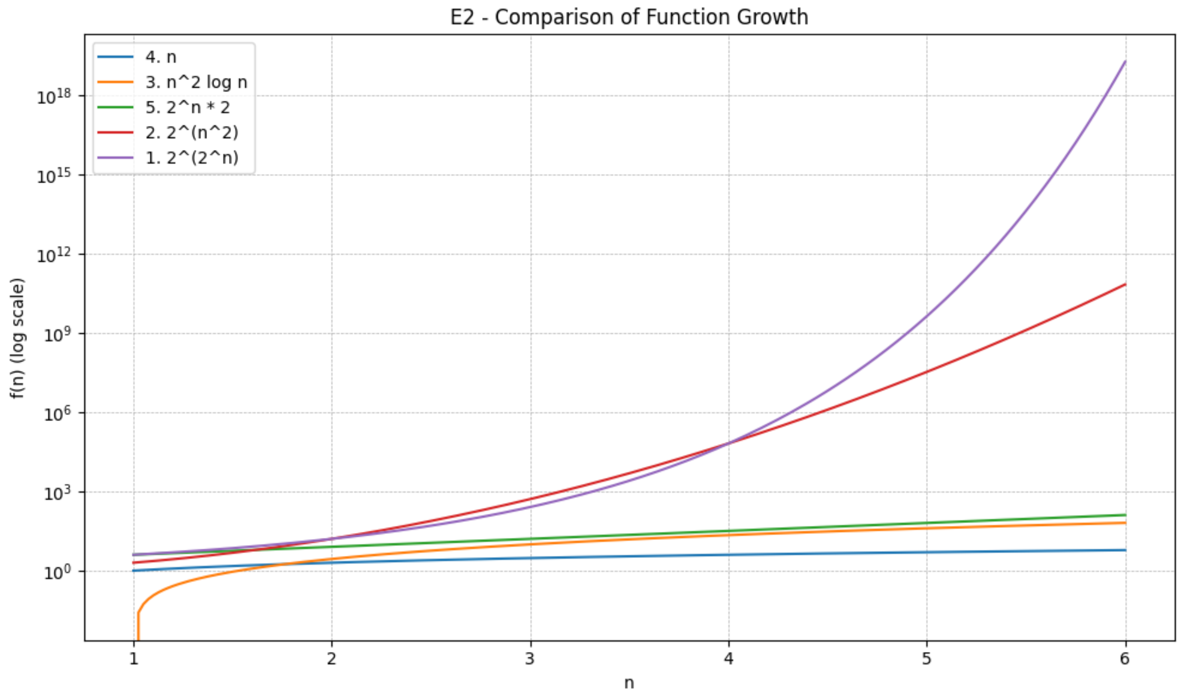

Figure 17: Graphical growth comparison of E2 expressions.

Conclusion

The comparative analysis of different algorithmic complexity functions provides valuable insight into their asymptotic behavior as the input size n increases. Through graphical visualization, several key observations can be made:

- **Slow-growing functions** such as $\log n$, $\frac{1}{n}$, $n \log n$ increase moderately, even for relatively large values of n. hese functions are typically associated with e**fficient and scalable algorithms**, well-suited for processing large datasets.
- In contrast, **exponential functions** like 2^n and π^n grow extremely rapidly. As soon as n exceeds a small threshold, these curves explode, making the corresponding algorithms **unusable in practice for large inputs**.
- Intermediate functions such as $n^{\log n}$, $10n + n^{3/2}$ and $2^{\log^2 n}$ offer a balance between performance and complexity, but require more context-specific analysis.

  The use of a logarithmic scale on the vertical axis was essential to make all curves visible on a single graph, smoothing out large differences in growth rates. This highlights the importance of carefully selecting algorithmic structures based on the complexity of their associated functions and the scale of data to be processed.