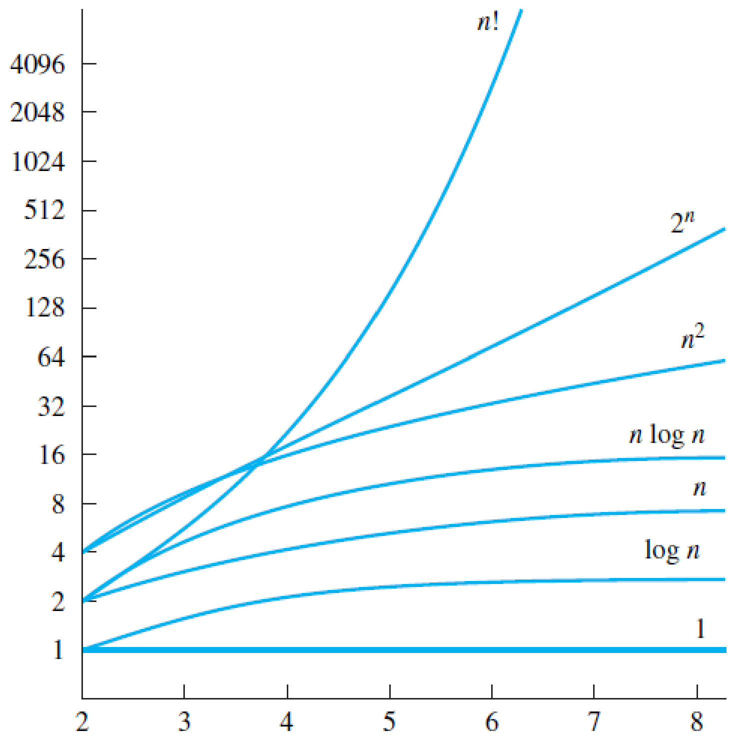
Terms, Concepts, and Examples

Recall: Big O notation is a standard way mathematicians and computer scientists use to describe how much time and how much memory is required for an algorithm to run.

We also saw a comparison of the growth of the functions.



Constant Complexity	$O(1)$
Logarithmic Complexity	$O(\log n)$
Linear Complexity	$O(n)$
Linearithmic Complexity	$O(n \log n)$
Quadratic Complexity	$O(n^2)$
Exponential Complexity	$O(b^n), b > 1$
Factorial Complexity	$O(n!)$

- Big O is all about the **Worst Case Scenario:** Consider searching the unsorted list $[1, 2, 8, 9, 3]$ for a specific number. The best case is searching for 1, an average case would be searching for 8, and the worst case is searching for 3. In big O analysis, we concern ourselves with worst cases.

  Example Count the number of operations (where an operation is an addition or a multiplication) used in this segment of an algorithm.

```
t = 0
for i in range(1,4):
```

```python
        for j in range(1,5):
            t = t+i*j
```

Solution: i runs through the values 1, 2, and 3. For each of those loops, j runs through the values 1, 2, 3, and 4. For example, when i= 1, there is 1 multiplication and one addition performed for each value of j. This gives 8 total operations (2 for each value of j).

If we sum this up across the three values of i, we have 24 operations in total. (8 for each value of i).

Video Example of Counting Operations

- **Constant Complexity** $O(1)$ - algorithm running time does not depend on size of input data. Complexity will always be a constant number.

  Python Tutor Example

- **Logarithmic Complexity -** $O(\log n)$ - the running time of the algorithm increases at most linearly as the number of inputs goes up exponentially. Intuitively, what tends to happen is

  - Each input can come from several possibilities
  - Only one will be chosen

  Python Tutor Example - while this is not $O(\log n)$ it shows an example of cutting the input size each time.

  The Binary Search algorithm has logarithmic complexity.

  Binary Search on Python

- **Linear Complexity** $O(n)$ - the running time of the algorithm increases at most linearly with the size of the input.

  Python Tutor Example

  The Linear Search algorithm has linear complexity.

  Linear Search with Python Tutor

- **Linearithmic Complexity** $O(n \log n)$ - this occurs when an operation is performed on each input such that each operation has logarithm time complexity. Mergesort is an algorithm that has linearithmic complexity (we will see this algorithm again in a later section).

- **Quadratic Complexity** $O(n^2)$ - the algorithm performs an at most linear time operation for each input.

  Python Tutor Example

  The Bubble Sort algorithm has quadratic complexity because each inputted value is, at worst, compared to all the other values.

  Bubble Sort on Python Tutor

- **Exponential Complexity -** $O(b^n), b > 1$ - the algorithm complexity increases by a constant multiple with each new input. For $O(2^n)$ the complexity doubles with each new input. We will revisit this when we cover recursion and trees.

- **Factorial Complexity -** $O(n!)$ - the complexity increases in a factorial way that depends on the number of inputs. We will also revisit this one.

  Video Example of Determining Big O of an Algorithm

  Python Tutor from Video Example

Practice Problems

1. Give a big O estimate for the number of operations, where an operation is an addition or a multiplication, used in this segment of an algorithm (ignoring comparisons used to test the conditions in the while loop).

```
i = 1
t = 0
while i <= n:
        t = t+i
        i = 2*i
```

2. Give a big O estimate for the number of comparisons used by the algorithm that determines the number of 1s in a list by examining each element of the list to determine whether it is a 1.

3. Jack has devised an algorithm that sorts through French text of n words and converts it into a pig latin document. Jacks algorithm takes $3n^2 + 3^n$ bit operations to handle an input text with n words. Suppose the computers in your business can handle one bit operation every nanosecond (1 nanosecond = 10^-9 seconds).

   (a) How many nanoseconds would it take Jack's algorithm to convert a text with 12 words on these computers?

   (b) How many HOURS would it take Jack's algorithm to convert a text with 50 words on these computers?