Algorithms and Data Structures

Lec02

Asymptotic Analysis

Dr. Mohammad Ahmad

Asymptotic Analysis

Outline:

we will look at:

- Justification for analysis
- Quadratic and polynomial growth
- Landau symbols
- Big-Θ as an equivalence relation
- Little-$\mathbf{o}$ as a weak ordering

Background

Suppose we have two algorithms, how can we tell which is better?

We could implement both algorithms, run them both

- Expensive and error prone

Preferably, we should analyze them mathematically

- *Algorithm analysis*

Asymptotic Analysis

In general, we will always analyze algorithms with respect to one or more variables

We will begin with one variable:

- The number of items n currently stored in an array or other data structure
- The number of items expected to be stored in an array or other data structure
- The dimensions of an $n \times n$ matrix

Examples with multiple variables:

- Dealing with n objects stored in m memory locations
- Multiplying a $k \times m$ and an $m \times n$ matrix
- Dealing with sparse matrices of size $n \times n$ with m non-zero entries

Maximum Value

For example, the time taken to find the largest object in an array of n random integers will take n operations

```
int find_max( int *array, int n ) {
    int max = array[0];

    for ( int i = 1; i < n; ++i ) {
        if ( array[i] > max ) {
            max = array[i];
        }
    }

    return max;
}
```

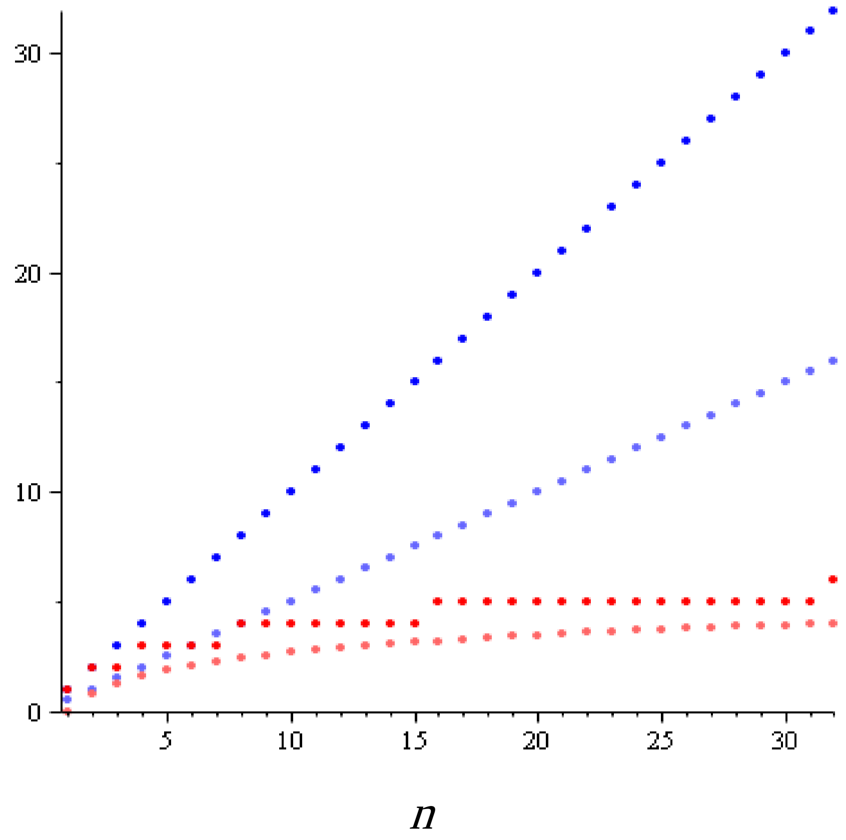
Linear and binary search

There are other algorithms which are significantly faster as the problem size increases

This plot shows maximum and average number of comparisons to find an entry in a sorted array of size n

– Linear search

– Binary search



Asymptotic Analysis

Given an algorithm:

- We need to be able to describe these values mathematically
- We need a systematic means of using the description of the algorithm together with the properties of an associated data structure
- We need to do this in a machine-independent way

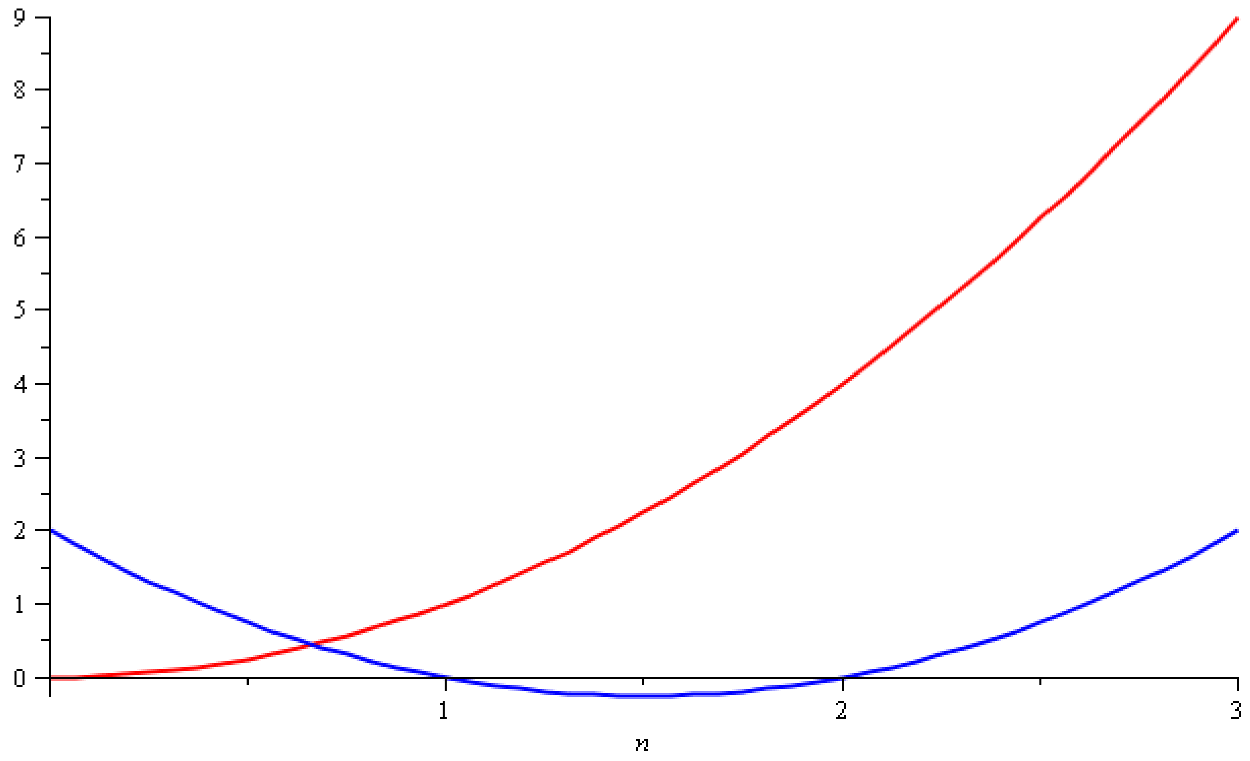
For this, we need Landau symbols and the associated asymptotic analysis

Quadratic Growth

Consider the two functions

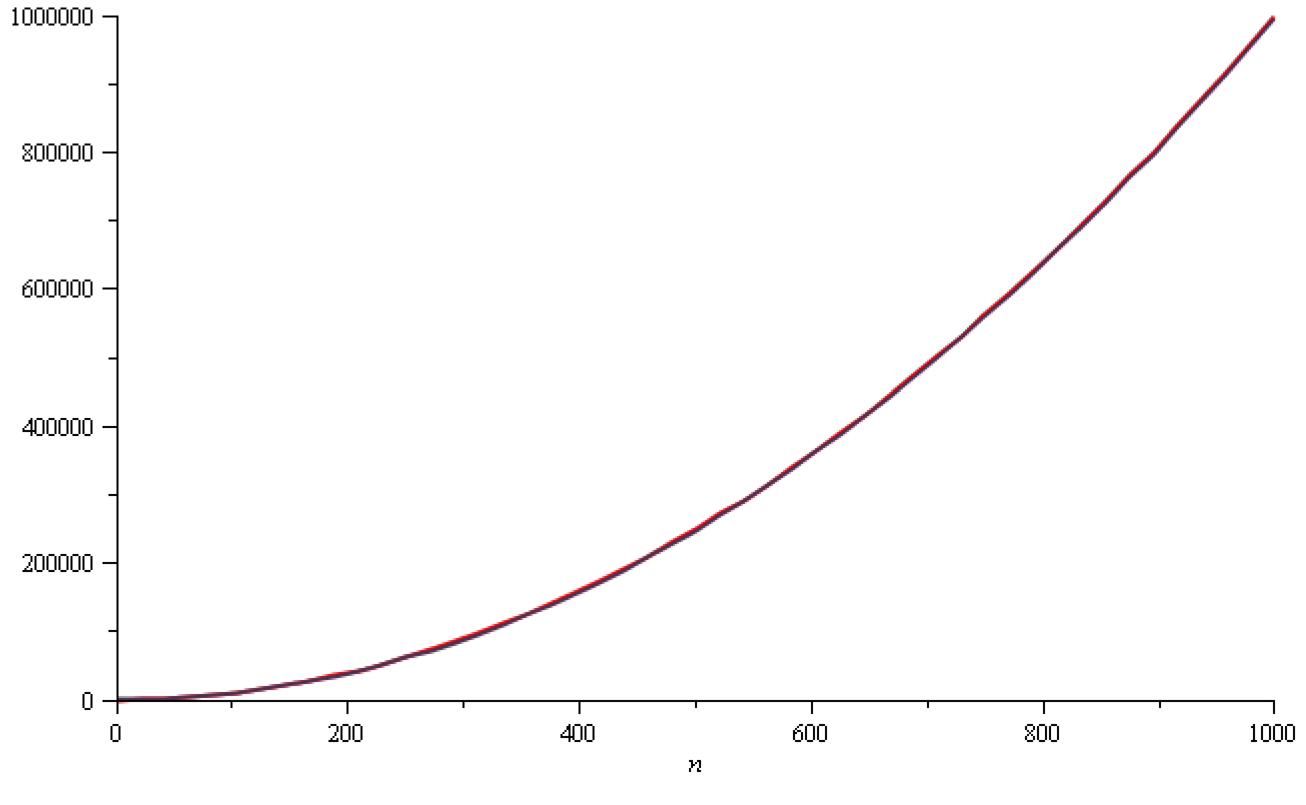
$f(n) = n^2$ and $g(n) = n^2 - 3n + 2$

Around $n = 0$, they look very different



Quadratic Growth

Yet on the range $n = [0, 1000]$, they are (relatively) indistinguishable:



Quadratic Growth

The absolute difference is large, for example,

$$f(1000) = 1\,000\,000$$

$$g(1000) = \quad 997\,002$$

but the relative difference is very small

$$\left|\frac{f(1000) - g(1000)}{f(1000)}\right| = 0.002998 < 0.3\%$$

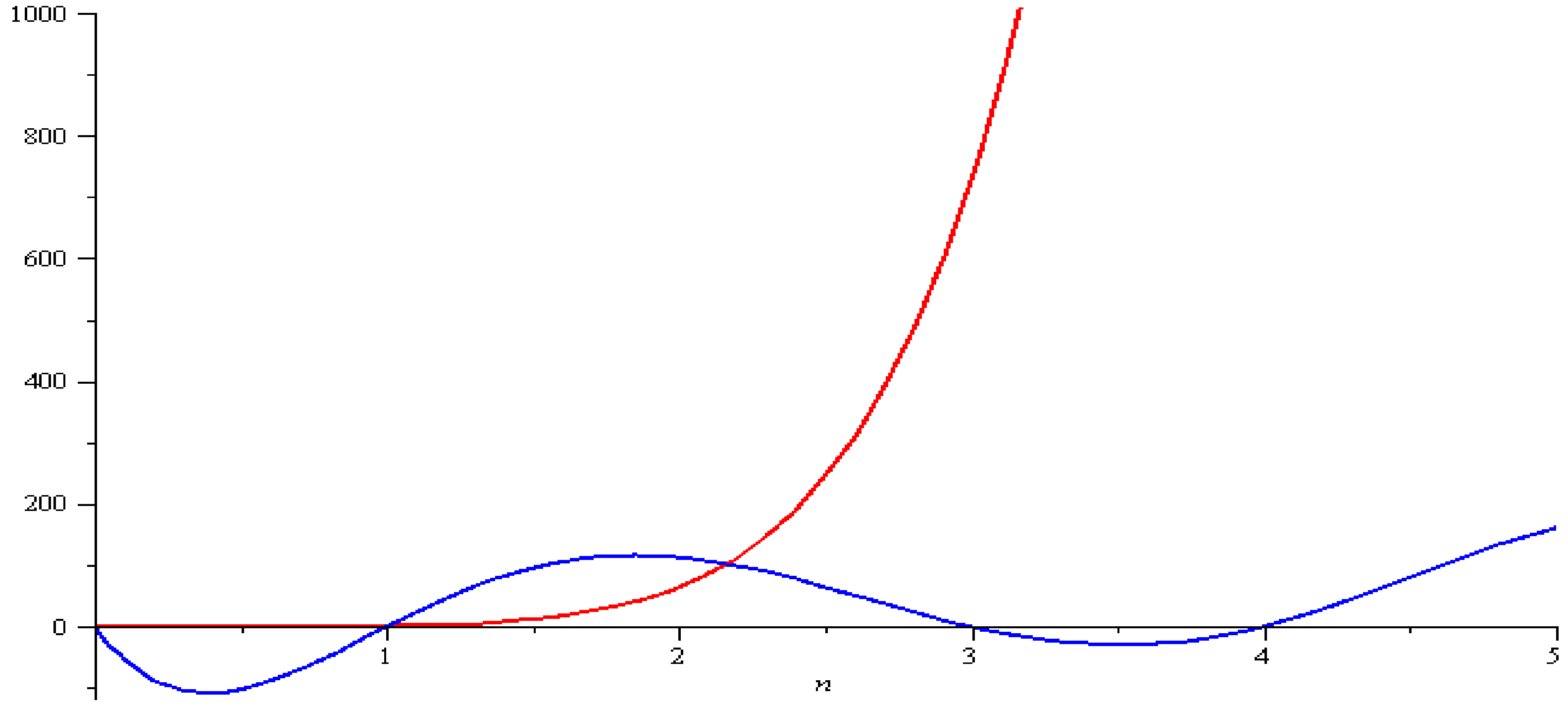
and this difference goes to zero as $n \to \infty$

Polynomial Growth

To demonstrate with another example,

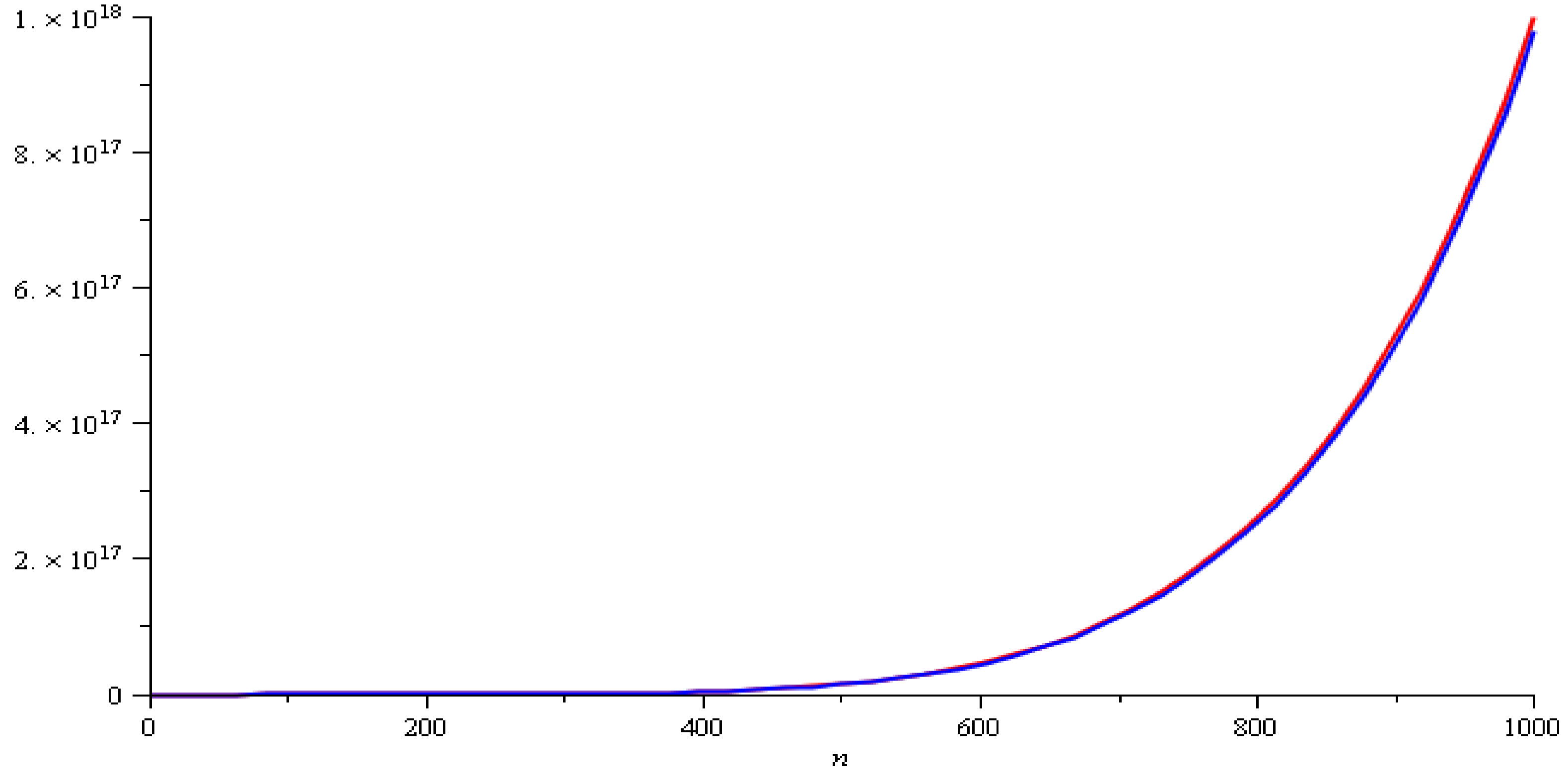
$f(n) = n^6$ and $g(n) = n^6 - 23n^5 + 193n^4 - 729n^3 + 1206n^2 - 648n$

Around $n = 0$, they are very different



Polynomial Growth

Still, around $n = 1000$, the relative difference is less than 3%



Polynomial Growth

The justification for both pairs of polynomials being similar is that, in both cases, they each had the same leading term:

n^2 in the first case, n^6 in the second

Suppose however, that the coefficients of the leading terms were different

- In this case, both functions would exhibit the same rate of growth, however, one would always be proportionally larger

Examples

We will now look at example:

- A comparison of selection sort and Merge sort.

```
public static void selectionSort(int[] arr)
{
    for (int i = 0; i < arr.length - 1; i++)
        for (int j = i + 1; j < arr.length; j++)
        {
            if (arr[j] < arr[i])
                {
                    int temp = arr[i];
                     arr[i] = arr[j];
                     arr[j] = temp;
                }
    }
}
```

```
MergeSort (Array[First..Last])
{
  If (Array contains only one element)
      Return Array
  else
   {
    Middle= ((Last + First)/2) rounded
down to the nearest integer
  MergeSort(Array(First..Middle))
  MergeSort(Array(Middle+1..Last))
ResultArray = Merge(LeftHalfArray,
RightHalfArray)
Return ResultArray
    }
}
```

Weak ordering

Consider the following definitions:

– We will consider two functions to be equivalent, $f \sim g$, if

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = c \quad \text{where} \quad 0 < c < \infty$$

– We will state that $f < g$ if $\lim_{n\to\infty}\frac{f(n)}{g(n)} = 0$

For functions we are interested in, these define a weak ordering

Weak ordering

Let $f(n)$ and $g(n)$ describe either the run-time of two algorithms

- If $f(n) \sim g(n)$, then it is always possible to improve the performance of one function over the other by purchasing a faster computer
- If $f(n) < g(n)$, then you can never purchase a computer fast enough so that the second function always runs in less time than the first

Note that for small values of *n*, it may be reasonable to use an algorithm that is asymptotically more expensive, but we will consider these on a one-by-one basis

Weak ordering

In general, there are functions such that

- If $f(n) \sim g(n)$, then it is always possible to improve the performance of one function over the other by purchasing a faster computer
- If $f(n) < g(n)$, then you can never purchase a computer fast enough so that the second function always runs in less time than the first

Note that for small values of n, it may be reasonable to use an algorithm that is asymptotically more expensive, but we will consider these on a one-by-one basis

Landau Symbols

A function $\mathrm{f}(n) = \mathbf{O}(\mathrm{g}(n))$ if there exists N and c such that

$$\mathrm{f}(n) < c\,\mathrm{g}(n)$$

whenever $n > N$

– The function $\mathrm{f}(n)$ has a rate of growth no greater than that of $\mathrm{g}(n)$

Landau Symbols

Before we begin, however, we will make some assumptions:

- Our functions will describe the time or memory required to solve a problem of size n
- We conclude we are restricting ourselves to certain functions:
  - They are defined for $n \geq 0$
  - They are strictly positive for all n
    - In fact, $\mathrm{f}(n) > c$ for some value $c > 0$
    - That is, any problem requires at least one instruction and byte
  - They are increasing (monotonic increasing)

Landau Symbols

Another Landau symbol is Θ

A function $f(n) = \Theta(g(n))$ if there exist positive N, c_1, and c_2 such that

$$c_1\, g(n) < f(n) < c_2\, g(n)$$

whenever $n > N$

– The function $f(n)$ has a rate of growth equal to that of $g(n)$

Landau Symbols

These definitions are often unnecessarily tedious

$$\lim_{n\to\infty}\frac{\mathrm{f}(n)}{\mathrm{g}(n)} = c \qquad \text{where} \quad 0 < c < \infty$$

Note, however, that if $\mathrm{f}(n)$ and $\mathrm{g}(n)$ are polynomials of the same degree with positive leading coefficients:

Landau Symbols

Suppose that $f(n)$ and $g(n)$ satisfy $\lim_{n\to\infty}\frac{f(n)}{g(n)} = c$

From the definition, this means given $c > \varepsilon > 0$ there exists an $N > 0$ such that $\left|\frac{f(n)}{g(n)} - c\right| < \varepsilon$ whenever $n > N$

That is, $c - \varepsilon < \frac{f(n)}{g(n)} < c + \varepsilon$

$$g(n)(c-\varepsilon) < f(n) < g(n)(c+\varepsilon)$$

Landau Symbols

However, the statement $\mathrm{g}(n)(c-\varepsilon) < \mathrm{f}(n) < \mathrm{g}(n)(c+\varepsilon)$ says that $\mathrm{f}(n) = \Theta(\mathrm{g}(n))$

Note that this only goes one way:

If $\lim_{n\to\infty} \frac{\mathrm{f}(n)}{\mathrm{g}(n)} = c$ where $0 < c < \infty$, it follows that

$\mathrm{f}(n) = \Theta(\mathrm{g}(n))$

Landau Symbols

We have a similar definition for **O**:

If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ where $0 \le c < \infty$, it follows that

$f(n) = \mathbf{O}(g(n))$

There are other possibilities we would like to describe:

If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, we will say $f(n) = \mathbf{o}(g(n))$

- The function $f(n)$ has a rate of growth less than that of $g(n)$, We would also like to describe the opposite cases:
- The function $f(n)$ has a rate of growth greater than that of $g(n)$
- The function $f(n)$ has a rate of growth greater than or equal to that of $g(n)$

Landau Symbols

We will at times use five possible descriptions

$$f(n) = \mathbf{o}(g(n)) \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) = \mathbf{O}(g(n)) \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

$$f(n) = \mathbf{\Theta}(g(n)) \qquad 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

$$f(n) = \mathbf{\Omega}(g(n)) \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$$

$$f(n) = \boldsymbol{\omega}(g(n)) \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Landau Symbols

For the functions we are interested in, it can be said that

$f(n) = \mathbf{O}(g(n))$ is equivalent to $f(n) = \mathbf{\Theta}(g(n))$ or $f(n) = \mathbf{o}(g(n))$

and

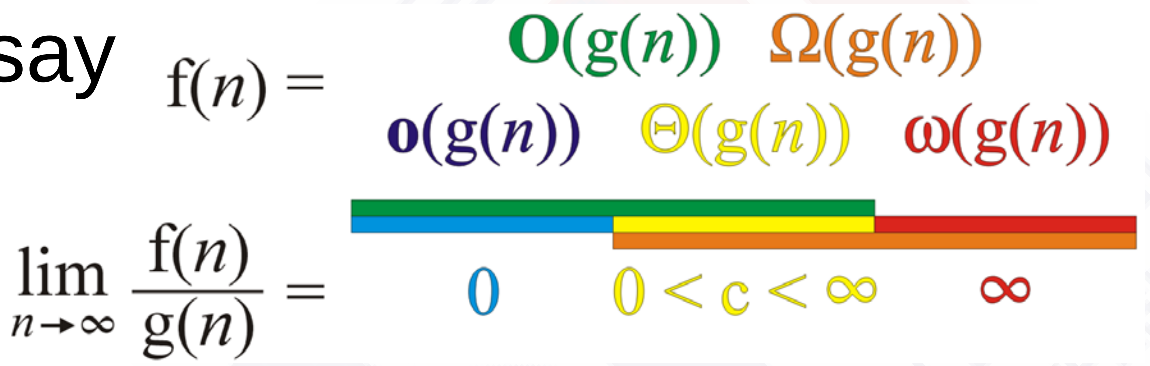
$f(n) = \mathbf{\Omega}(g(n))$ is equivalent to $f(n) = \mathbf{\Theta}(g(n))$ or $f(n) = \boldsymbol{\omega}(g(n))$

Landau Symbols

Graphically, we can summarize these as follows:

We say

if



$f(n) = $
$O(g(n))$ $\Omega(g(n))$
$o(g(n))$ $\Theta(g(n))$ $\omega(g(n))$
$\lim_{n \to \infty} \frac{f(n)}{g(n)} = $
0
$0 < c < \infty$
∞

Landau Symbols

Some other observations we can make are:

$$f(n) = \mathbf{\Theta}(g(n)) \Leftrightarrow g(n) = \mathbf{\Theta}(f(n))$$

$$f(n) = \mathbf{O}(g(n)) \Leftrightarrow g(n) = \mathbf{\Omega}(f(n))$$

$$f(n) = \mathbf{o}(g(n)) \Leftrightarrow g(n) = \boldsymbol{\omega}(f(n))$$

Big-Θ as an Equivalence Relation

If we look at the first relationship, we notice that $f(n) = \Theta(g(n))$ seems to describe an equivalence relation:

1. $f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$
2. $f(n) = \Theta(f(n))$
3. If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$, it follows that $f(n) = \Theta(h(n))$

Consequently, we can group all functions into equivalence classes, where all functions within one class are big-theta Θ of each other

Big-Θ as an Equivalence Relation

For example, all of

n^2 $100000\,n^2 - 4\,n + 19$ $n^2 + 1000000$

$323\,n^2 - 4\,n\ln(n) + 43\,n + 10$ $42n^2 + 32$

$n^2 + 61\,n\ln^2(n) + 7n + 14\ln^3(n) + \ln(n)$

are big-Θ of each other

E.g., $42n^2 + 32 = \Theta(\,323\,n^2 - 4\,n\ln(n) + 43\,n + 10\,)$

Big-Θ as an Equivalence Relation

The most common classes are given names:

$\Theta(1)$	constant
$\Theta(\ln(n))$	logarithmic
$\Theta(n)$	linear
$\Theta(n \ln(n))$	“n log n”
$\Theta(n^2)$	quadratic
$\Theta(n^3)$	cubic
$2^n, e^n, 4^n, \ldots$	exponential

Logarithms and Exponentials

Recall that all logarithms are scalar multiples of each other

- Therefore $\log_b(n) = \Theta(\ln(n))$ for any base b

Alternatively, there is no single equivalence class for exponential functions:

- If $1 < a < b$,

$$\lim_{n\to\infty} \frac{a^n}{b^n} = \lim_{n\to\infty} \left(\frac{a}{b}\right)^n = 0$$

- Therefore $a^n = \mathbf{o}(b^n)$

However, we will see that it is almost universally undesirable to have an exponentially growing function!

Logarithms and Exponentials

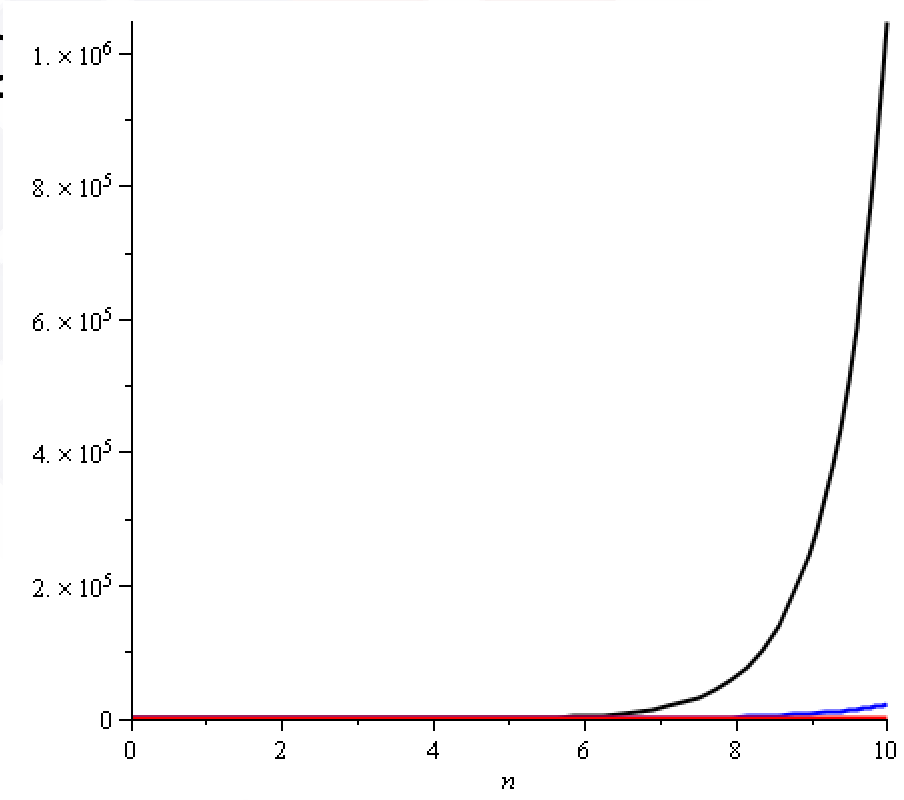
Plotting 2^n, e^n, and 4^n on the range $[1, 10]$ already shows how significantly different the functions g

Note:

$2^{10} = 1024$

$e^{10} \approx 22\,026$

$4^{10} = 1\,048\,576$



Little-**o** as a Weak Ordering

We can show that, for example

$$\ln(n) = \mathbf{o}(n^p)$$

for any $p > 0$

Proof: Using l'Hôpital's rule, we have

$$\lim_{n\to\infty}\frac{\ln(n)}{n^p} = \lim_{n\to\infty}\frac{1/n}{pn^{p-1}} = \lim_{n\to\infty}\frac{1}{pn^p} = \frac{1}{p}\lim_{n\to\infty} n^{-p} = 0$$

Conversely, $1 = \mathbf{o}(\ln(n))$

Little-$\mathbf{o}$ as a Weak Ordering

Other observations:

– If p and q are real positive numbers where $p < q$, it follows that

$$n^p = \mathbf{o}(n^q)$$

– For example, matrix-matrix multiplication is $\Theta(n^3)$ but a refined algorithm is $\Theta(n^{\lg(7)})$ where $\lg(7) \approx 2.81$

– Also, $n^p = \mathbf{o}(\ln(n)n^p)$, but $\ln(n)n^p = \mathbf{o}(n^q)$

- n^p has a slower rate of growth than $\ln(n)n^p$, but
- $\ln(n)n^p$ has a slower rate of growth than n^q for $p < q$

Little-**o** as a Weak Ordering

If we restrict ourselves to functions $f(n)$ which are $\Theta(n^p)$ and $\Theta(\ln(n)n^p)$, we note:

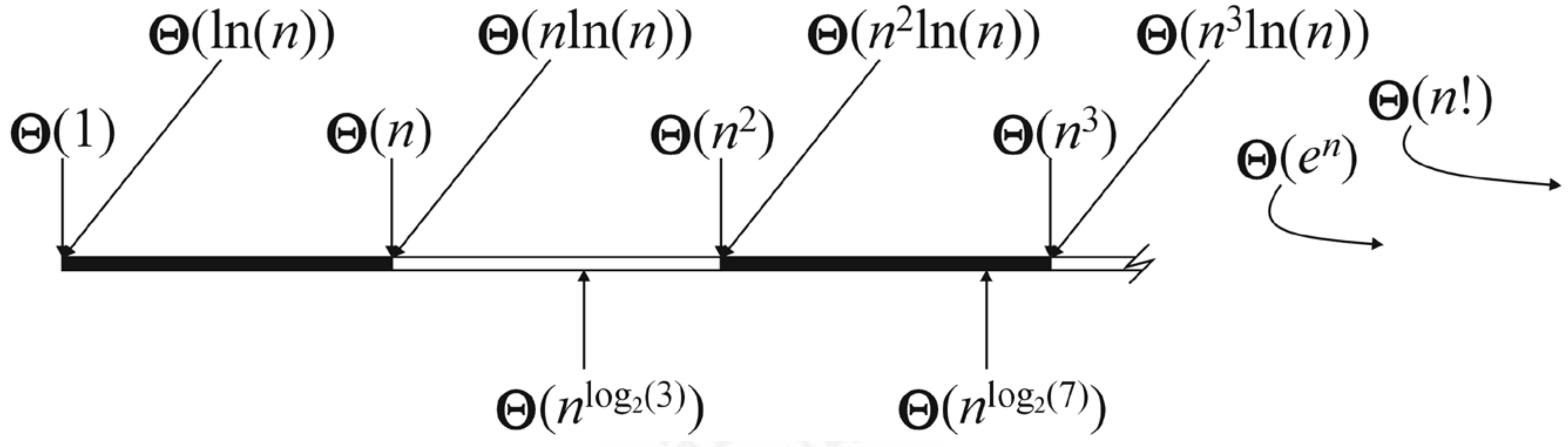
- It is never true that $f(n) = \mathbf{o}(f(n))$
- If $f(n) \neq \Theta(g(n))$, it follows that either

  $f(n) = \mathbf{o}(g(n))$ or $g(n) = \mathbf{o}(f(n))$
- If $f(n) = \mathbf{o}(g(n))$ and $g(n) = \mathbf{o}(h(n))$, it follows that $f(n) = \mathbf{o}(h(n))$

This defines a weak ordering!

Little-o as a Weak Ordering

Graphically, we can shown this relationship by marking these against the real line



Algorithms Analysis

We will use Landau symbols to describe the complexity of algorithms

- E.g., adding a list of n doubles will be said to be a $\Theta(n)$ algorithm

An algorithm is said to have *polynomial time complexity* if its run-time may be described by $\mathrm{O}(n^d)$ for some fixed $d \geq 0$

- We will consider such algorithms to be *efficient*

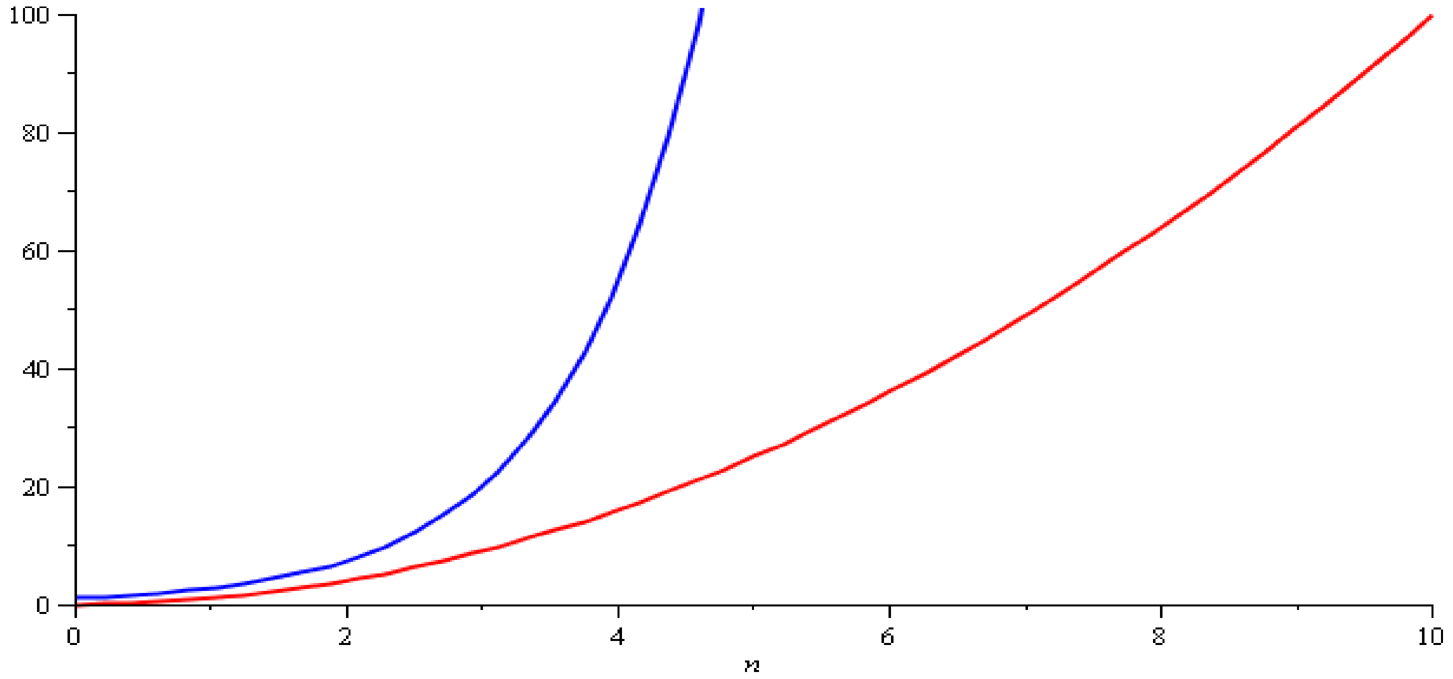
Problems that have no known polynomial-time algorithms are said to be *intractable*

- Traveling salesman problem: find the shortest path that visits n cities
- Best run time: $\Theta(n^2 2^n)$

Algorithm Analysis

In general, you don't want to implement exponential-time or exponential-memory algorithms

- Warning: don't call a quadratic curve "exponential", either...please



100
80
60
40
20
0
0
2
4
6
8
10
n