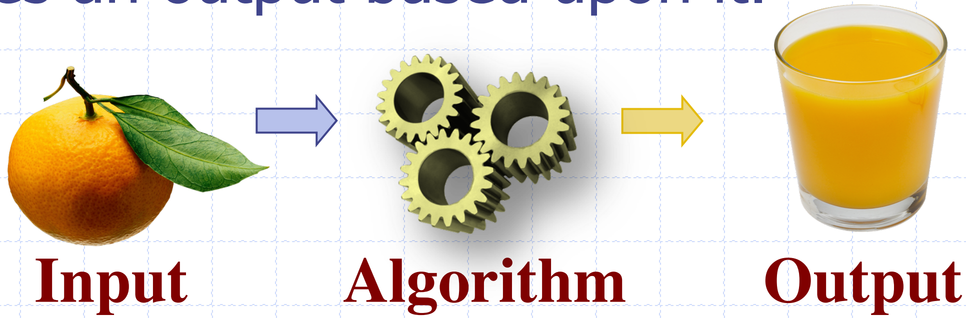
Lecture 2
Math overview

CS 161 Design and Analysis of Algorithms

Ioannis Panageas

Algorithms and Data Structures

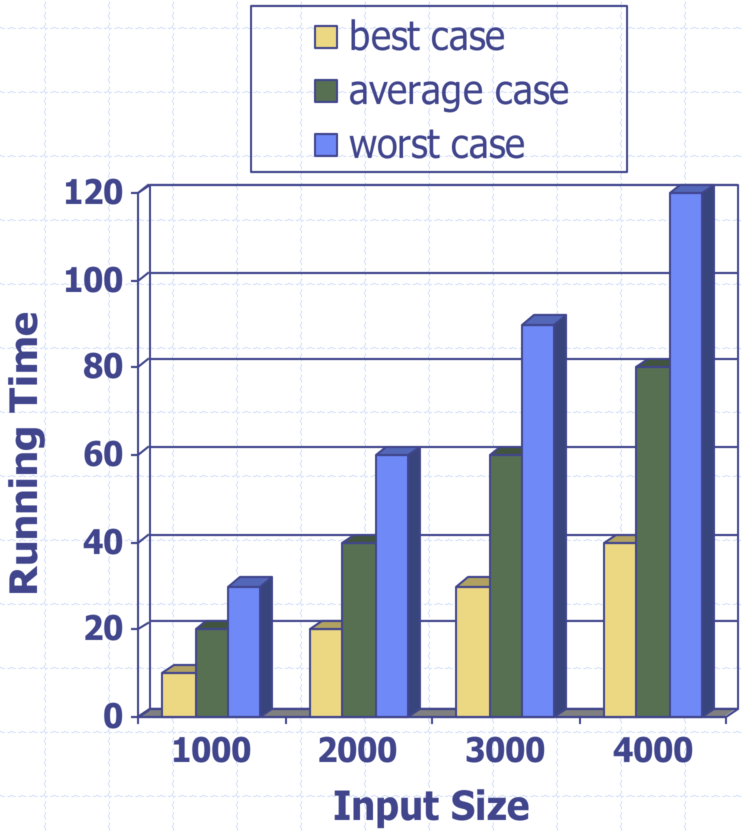
- An **algorithm** is a step-by-step procedure for performing some task in a finite amount of time.
  - Typically, an algorithm takes input data and produces an output based upon it.



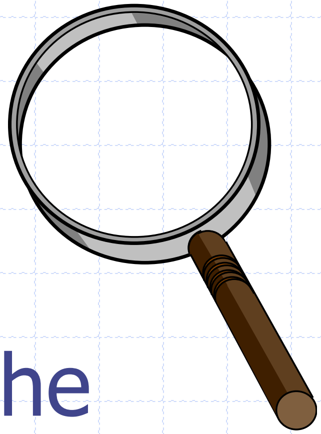
- A **data structure** is a systematic way of organizing and accessing data.

Running Time

- Most algorithms transform input objects into output objects.
- The running time of an algorithm typically grows with the input size.
- Average case time is often difficult to determine.
- We focus primarily on the **worst case running time**.
  - Easier to analyze
  - Crucial to applications such as games, finance and robotics



Theoretical Analysis



- Uses a high-level description of the algorithm instead of an implementation
- Characterizes running time as a function of the input size, n
- Takes into account all possible inputs
- Allows us to evaluate the speed of an algorithm independent of the hardware/software environment

Pseudocode

- High-level description of an algorithm
- More structured than English prose
- Less detailed than a program
- Preferred notation for describing algorithms
- Hides program design issues

Pseudocode Details

- Control flow
  - **if** … **then** … [**else** …]
  - **while** … **do** …
  - **repeat** … **until** …
  - **for** … **do** …
  - Indentation replaces braces
- Method declaration

  Algorithm *method* (*arg* [, *arg*…])

  Input …

  Output …

- Method call

  method (*arg* [, *arg*…])
- Return value

  return *expression*
- Expressions:

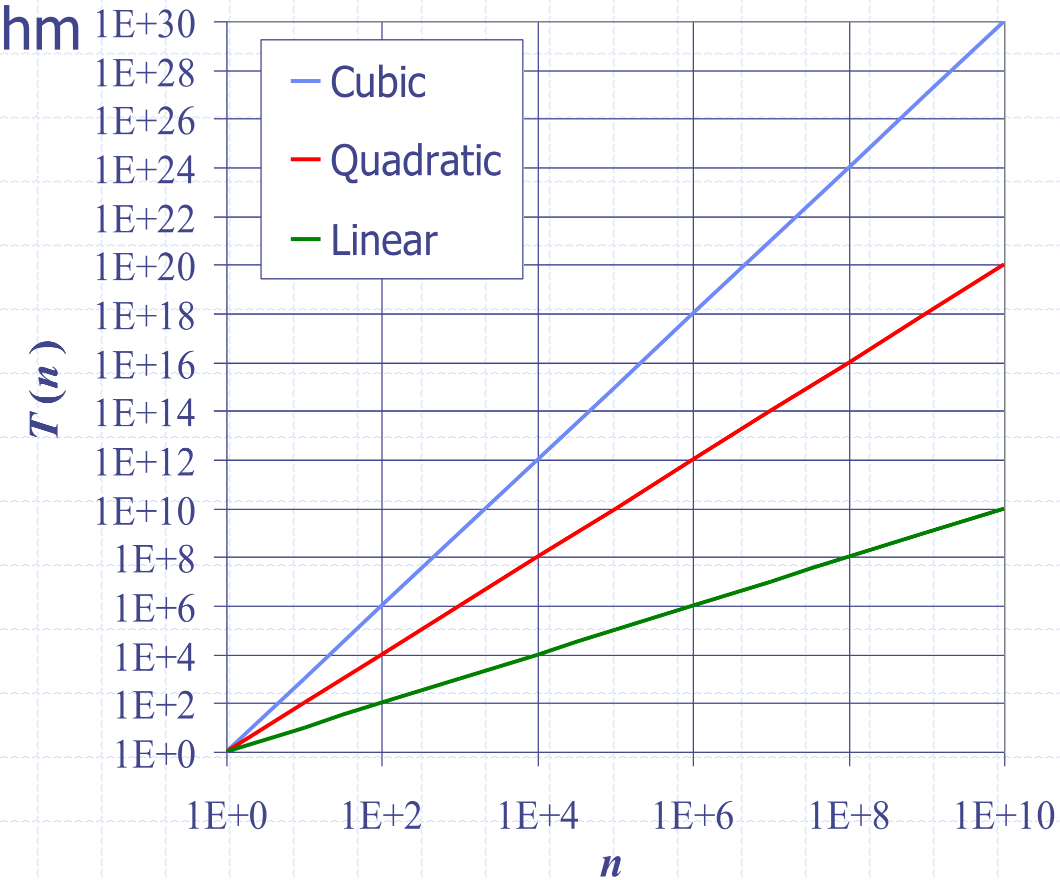
  $\leftarrow$ Assignment

  $=$ Equality testing

  n^2 Superscripts and other mathematical formatting allowed

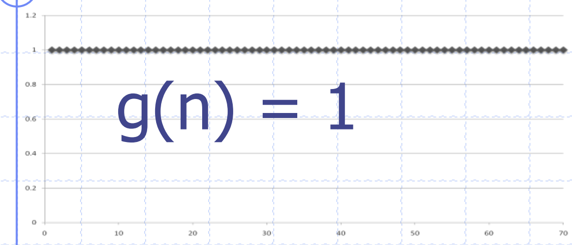
Seven Important Functions

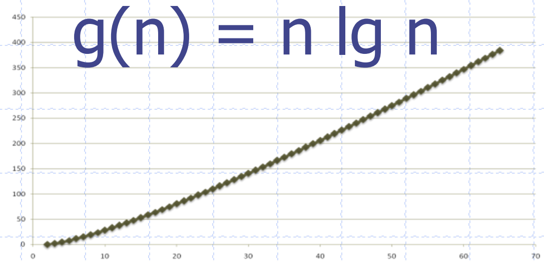
- Seven functions that often appear in algorithm analysis:
  - Constant ≈ 1
  - Logarithmic ≈ $\log n$
  - Linear ≈ n
  - N-Log-N ≈ $n \log n$
  - Quadratic ≈ n^2
  - Cubic ≈ n^3
  - Exponential ≈ 2^n
- In a log-log chart, the slope of the line corresponds to the growth rate

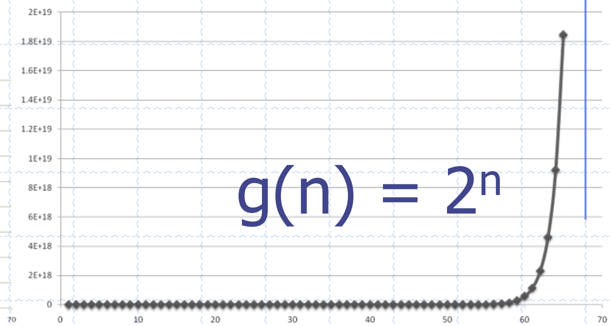


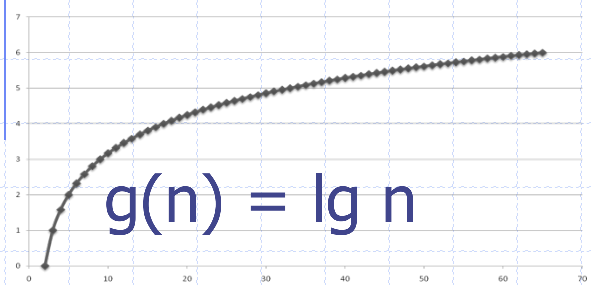
Functions Graphed Using "Normal" Scale

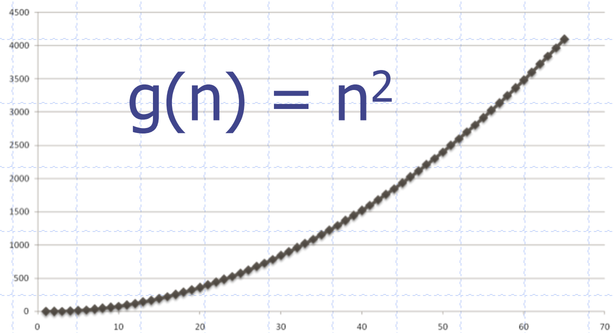
Slide by Matt Stallmann included with permission.

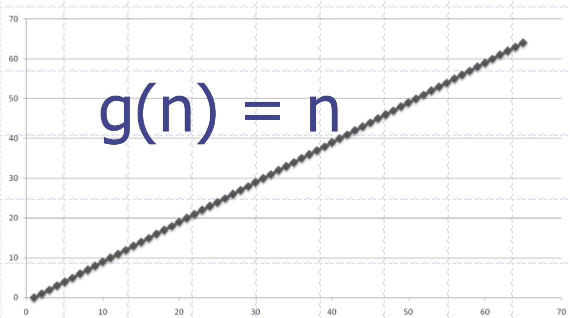


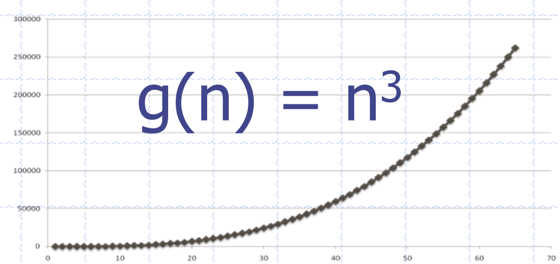












Primitive Operations

- Basic computations performed by an algorithm
- Identifiable in pseudocode
- Largely independent from the programming language
- Exact definition not important

- Examples:
  - Evaluating an expression
  - Assigning a value to a variable
  - Indexing into an array
  - Calling a method

Counting Primitive Operations

- Example: By inspecting the pseudocode, we can determine the maximum number of primitive operations executed by an algorithm, as a function of the input size

Algorithm arrayMax(A, n):

    Input: An array A storing $n \geq 1$ integers.

    Output: The maximum element in A.

  $\mathit{currentMax} \leftarrow A[0]$

  for $i \leftarrow 1$ **to** $n - 1$ **do**

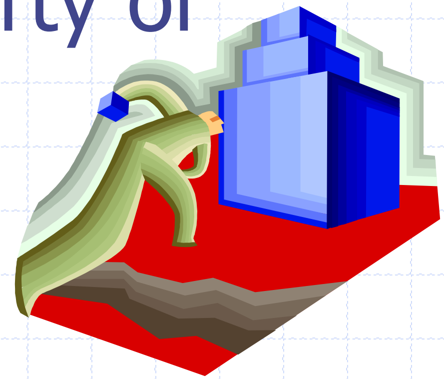
      if $\mathit{currentMax} < A[i]$ **then**

          $\mathit{currentMax} \leftarrow A[i]$

  return $\mathit{currentMax}$

Growth Rate of Running Time

- Changing the hardware/ software environment
  - Affects $T(n)$ by a constant factor, but
  - Does not alter the growth rate of $T(n)$
- The linear growth rate of the running time $T(n)$ is an intrinsic property of algorithm arrayMax



Slide by Matt Stallmann included with permission.

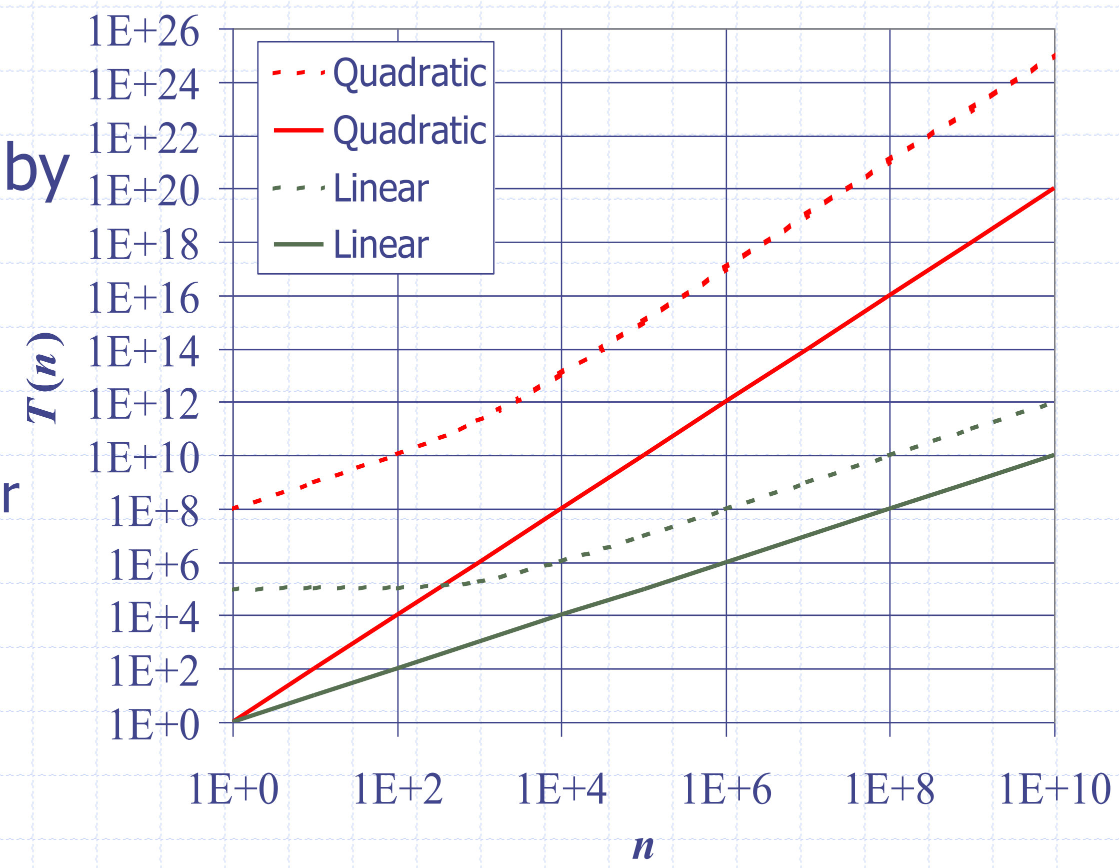
Why Growth Rate Matters

if runtime is...	time for n + 1	time for 2 n	time for 4 n
$c \lg n$	$c \lg (n + 1)$	$c (\lg n + 1)$	$c(\lg n + 2)$
$c\, n$	$c\, (n + 1)$	$2c\, n$	$4c\, n$
$c\, n \lg n$	$\sim c\, n \lg n + c\, n$	$2c\, n \lg n + 2cn$	$4c\, n \lg n + 4cn$
$c\, n^2$	$\sim c\, n^2 + 2c\, n$	**$4c\, n^2$**	$16c\, n^2$
$c\, n^3$	$\sim c\, n^3 + 3c\, n^2$	$8c\, n^3$	$64c\, n^3$
$c\, 2^n$	$c\, 2^{n+1}$	$c\, 2^{2n}$	$c\, 2^{4n}$

runtime quadruples when problem size doubles

Constant Factors

- The growth rate is minimally affected by
  - constant factors or
  - lower-order terms
- Examples
  - $10^2 n + 10^5$ is a linear function
  - $10^5 n^2 + 10^8 n$ is a quadratic function



Asymptotic Algorithm Analysis

- The asymptotic analysis of an algorithm determines the running time in big-Oh notation
- To perform the asymptotic analysis
  - We find the worst-case number of primitive operations executed as a function of the input size
  - We express this function with big-Oh notation
- Example:
  - We say that algorithm arrayMax "runs in $O(n)$ time"
- Since constant factors and lower-order terms are eventually dropped anyhow, we can disregard them when counting primitive operations

Big-Oh Rules

- If is $f(n)$ a polynomial of degree d, then $f(n)$ is $O(n^d)$, i.e.,
  1. Drop lower-order terms
  2. Drop constant factors
- Use the smallest possible class of functions
  - Say "$2n$ is $O(n)$" instead of "$2n$ is $O(n^2)$"
- Use the simplest expression of the class
  - Say "$3n + 5$ is $O(n)$" instead of "$3n + 5$ is $O(3n)$"

Analyzing Recursive Algorithms

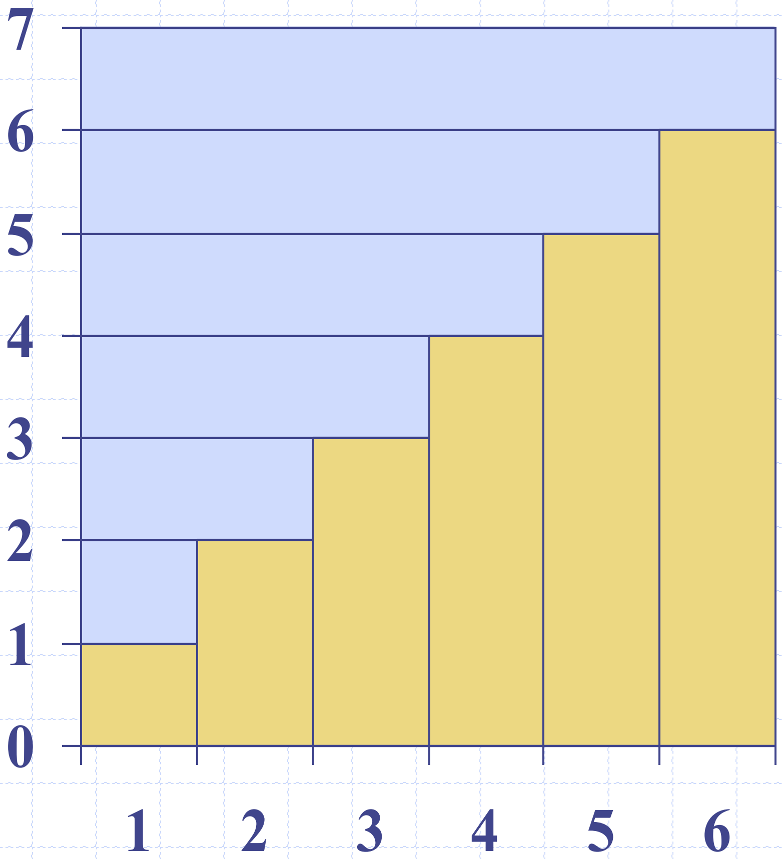
- Use a function, T(*n*), to derive a **recurrence relation** that characterizes the running time of the algorithm in terms of smaller values of *n*.

Algorithm recursiveMax(A, n):
Input: An array A storing $n \geq 1$ integers.
Output: The maximum element in A.
if $n = 1$ **then**
    return $A[0]$
return $\max\{\text{recursiveMax}(A, n-1),\ A[n-1]\}$

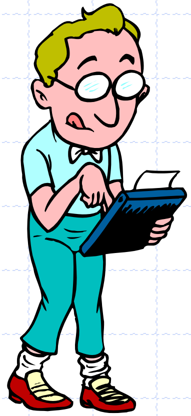
$$T(n) = \begin{cases} 3 & \text{if } n = 1 \\ T(n-1) + 7 & \text{otherwise,} \end{cases}$$

Arithmetic Progression

- Assume the running time of P is $O(1 + 2 + \ldots + n)$
- The sum of the first n integers is $n(n + 1) / 2$
  - There is a simple visual proof of this fact
- Thus, algorithm P runs in $O(n^2)$ time



Math you need to Review



- Summations
- Powers
- Logarithms
- Proof techniques
- Basic probability

- Properties of powers:
  - $a^{(b+c)} = a^b a^c$
  - $a^{bc} = (a^b)^c$
  - $a^b / a^c = a^{(b-c)}$
  - $b = a^{\log_a b}$
  - $b^c = a^{c*\log_a b}$
- Properties of logarithms:
  - $\log_b(xy) = \log_b x + \log_b y$
  - $\log_b (x/y) = \log_b x - \log_b y$
  - $\log_b xa = a\log_b x$
  - $\log_b a = \log_x a / \log_x b$

O ("big oh")

Informally:

- $g \in O(f)$ if g is bounded above by a constant multiple of f (for sufficiently large values of n).
- $g \in O(f)$ if "g grows no faster than (a constant multiple of) f."
- $g \in O(f)$ if the ratio g/f is bounded above by a constant (for sufficiently values of n).

O ("big oh")

Formally:

- $g \in O(f)$ if and only if:

$$\exists_{C>0}\, \exists_{n_0>0}\, \forall_{n>n_0}\; g(n) \leq C \cdot f(n).$$

- Equivalently: $g \in O(f)$ if and only if:

$$\exists_{C>0}\, \exists_{n_0>0}\, \forall_{n>n_0}\; \frac{g(n)}{f(n)} \leq C.$$

- Sometimes we write: $g = O(f)$ rather than $g \in O(f)$

Examples of O-notation:

Example 1: $f(n) = n$, $g(n) = 1000n$: $g \in O(f)$.

Proof: Let $C = 1000$. Then $g(n) \leq C \cdot f(n)$ for all n.

Examples of O-notation:

Example 2: $f(n) = n^2$, $g(n) = n^{3/2}$: $g \in O(f)$.

Proof: $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$.
Hence for any $C > 0$ the ratio is less than C as long as n is sufficiently large.(Of course, how large n must be to be "sufficiently large" depends on C).

Alternate Proof: If $n \geq 1$, $n^{1/2} \geq 1$, so $n^{3/2} \leq n^2$.
Hence we can choose $C = 1$ and $n_0 = 1$.

Examples of O-notation:

Example 3: $f(n) = n^3$, $g(n) = n^4$: $g \notin O(f)$.

Proof: $\lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$.
Hence there is no $C > 0$ such that $g(n) \leq C \cdot f(n)$ for sufficiently large n.

Examples of O-notation:

Example 4: $f(n) = n^2$, $g(n) = 5n^2 + 23n + 2$: $g \in O(f)$.

Proof: If $n \geq 1$, then $n \leq n^2$ and $1 \leq n^2$. Hence:

$$\begin{aligned} g(n) &= 5n^2 + 23n + 2 \\ &\leq 5n^2 + 23n^2 + 2n^2 \\ &\leq 30n^2 \\ &= 30f(n) \end{aligned}$$

So we can take $C = 30$, $n_0 = 1$.

More asymptotic notation: o ("little oh"), Ω ("big Omega")

- o ('little oh"):

$$g \in o(f) \quad \text{if and only if } \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0.$$

- Ω ("big Omega") (or just "Omega")

$$g \in \Omega(f) \quad \text{if and only if} \quad \exists_{C>0}\, \exists_{n_0>0}\, \forall_{n>n_0}\, g(n) \geq C \cdot f(n).$$

Equivalently:

$$g \in \Omega(f) \quad \text{if and only if} \quad \exists_{C>0}\, \exists_{n_0>0}\, \forall_{n>n_0}\, \frac{g(n)}{f(n)} \geq C.$$

One more definition: Θ ("Theta")

- $g \in \Theta(f)$ if and only if:

$$g \in O(f) \text{ and } g \in \Omega(f).$$

- Equivalently, $g \in \Theta(f)$ if and only if:

$$\exists_{C_1 > 0}\, \exists_{C_2 > 0}\, \exists_{n_0 > 0}\, \forall_{n > n_0}\; C_1 \leq \frac{g(n)}{f(n)} \leq C_2.$$

Examples of Asymptotic notation

Example 1: $f(n) = n$, $g(n) = 1000n$.

$g \in \Omega(f)$, $g \in \Theta(f)$

To see that $g \in \Omega(f)$, we can take $C = 1$.

Then $g(n) = 1000 \cdot n > 1 \cdot n = C \cdot f(n)$.

To see that $g \in \Theta(f)$, we could argue that $g \in O(f)$ (shown earlier) and $g \in \Omega(f)$ (shown above).

Or we can take $C_1 = 1$, $C_2 = 1000$. Then

$$C_1 \leq \frac{g(n)}{f(n} \leq C_2.$$

Examples of Asymptotic notation

Example 2: $f(n) = n^2$, $g(n) = n^{3/2}$:

$g \in o(f)$

Because $\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$.

Examples of Asymptotic notation

Example 3: $f(n) = n^3$, $g(n) = n^4$:

$g \in \Omega(f)$

Because $\lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$, so we can choose any C we want.

Examples of Asymptotic notation

Example 4: $f(n) = n^2$, $g(n) = 5n^2 - 23n + 2$:

$g \in \Omega(f)$.

Proof: If $n \geq 23$, then $23n \leq n^2$. Hence if $n \geq 23$:

$$\begin{aligned} g(n) &= 5n^2 - 23n + 2 \\ &\geq 5n^2 - n^2 \\ &\geq 4n^2 \\ &= 4f(n) \end{aligned}$$

So we can take $C = 4$, $n_0 = 23$.

Another Example

Example 5: $\ln n = o(n)$

Proof:

Examine the ratio $\frac{\ln n}{n}$ as $n \to \infty$.

If we try to evaluate the limit directly, we obtain the "indeterminate form" $\frac{\infty}{\infty}$.

We need to apply L'Hôpital's rule (from calculus).

Example 5, continued: $\ln n = o(n)$

L'Hôpital's rule: If the ratio of limits

$$\frac{\lim_{n\to\infty} g(n)}{\lim_{n\to\infty} f(n)}$$

is an indeterminate form (i.e., ∞/∞ or $0/0$), then

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{g'(n)}{f'(n)}$$

where f' and g' are, respectively, the derivatives of f and g.

Example 5, continued: $\ln n = o(n)$

Let $f(n) = n$, $g(n) = \ln n$.

Then $f'(n) = 1$, $g'(n) = 1/n$.

By L'Hôpital's rule:

$$\begin{aligned}
\lim_{n\to\infty} \frac{g(n)}{f(n)} &= \lim_{n\to\infty} \frac{g'(n)}{f'(n)} \\
&= \lim_{n\to\infty} \frac{1/n}{1} \\
&= \lim_{n\to\infty} \frac{1}{n} \\
&= 0.
\end{aligned}$$

Hence $g(n) = o(f(n))$.

Math background

- Sums, Summations
- Logarithms, Exponents Floors, Ceilings, Harmonic Numbers
- Proof Techniques
- Basic Probability

Sums, Summations

- Summation notation:

$$\sum_{i=a}^{b} f(i) = f(a) + f(a+1) + \cdots + f(b).$$

- Special cases:
  - What if $a = b$? $f(a)$
  - What if $a > b$? 0
- If $S = \{s_1, \ldots, s_n\}$ is a finite set:

$$\sum_{x \in S} f(x) = f(s_1) + f(s_2) + \cdots + f(s_n).$$

Geometric sum

- Geometric sum:

$$\sum_{i=0}^{n} a^i = 1 + a^1 + a^2 + \cdots + a^n = \frac{1 - a^{n+1}}{1 - a},$$

provided that $a \neq 1$.

- Previous formula holds for $a = 0$ because $a^0 = 1$ even when $a = 0$.
- Special case of geometric sum:

$$\sum_{i=0}^{n} 2^i = 1 + 2 + 4 + 8 + \cdots + 2^n = 2^{n+1} - 1.$$

Infinite Geometric sum

- From the previous slide:

$$\sum_{i=0}^{n} a^i = 1 + a^1 + a^2 + \cdots + a^n = \frac{1 - a^{n+1}}{1 - a},$$

provided that $a \neq 1$.

- If $|a| < 1$, we can take the limit as $n \to \infty$:

$$\sum_{i=0}^{\infty} a^i = 1 + a^1 + a^2 + \cdots = \frac{1}{1 - a},$$

- Special case of infinite geometric sum:

$$\sum_{i=0}^{\infty} \frac{1}{2^i} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 2.$$

Other Summations

- Sum of first n integers

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} = \Theta\left(n^2\right)$$

- Sum of first n squares

$$\sum_{i=1}^{n} i^2 = 1 + 4 + 9 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} = \Theta\left(n^3\right)$$

- In general, for any fixed positive integer k:

$$\sum_{i=1}^{n} i^k = 1 + 2^k + 3^k + \cdots + n^k = \Theta\left(n^{k+1}\right)$$

Logarithms

Definition: $\log_b x = y$ if and only if $b^y = x$.

Some useful properties:

1. $\log_b 1 = 0$.
2. $\log_b b^a = a$.
3. $\log_b(xy) = \log_b x + \log_b y$.
4. $\log_b(x^a) = a \log_b x$.
5. $x^{\log_b y} = y^{\log_b x}$.
6. $\log_x b = \frac{1}{\log_b x}$.
7. $\log_a x = \frac{\log_b x}{\log_b a}$.
8. $\log_a x = (\log_b x)(\log_a b)$.

Floors and ceilings

- $\lfloor x \rfloor$ = largest integer $\leq x$. (Read as Floor of x)
- $\lceil x \rceil$= smallest integer $\geq x$ (Read as Ceiling of x)

Factorials

- $n! = 1 \cdot 2 \cdots n$
- $n!$ represents the number of distinct permutations of n objects.

1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

Combinations

$\binom{n}{k}$ = The number of different ways of choosing k objects from a collection of n objects. (Pronounced "n choose k".)

Example: $\binom{5}{2} = 10$

$$\begin{array}{ccccc} \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} \\ \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \end{array}$$

Formula: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

Special cases: $\binom{n}{0} = 1$, $\binom{n}{1} = n$, $\binom{n}{2} = \frac{n(n-1)}{2}$

Harmonic Numbers

The nth Harmonic number is the sum:

$$H_n = \sum_{i=1}^{n} \frac{1}{i}$$

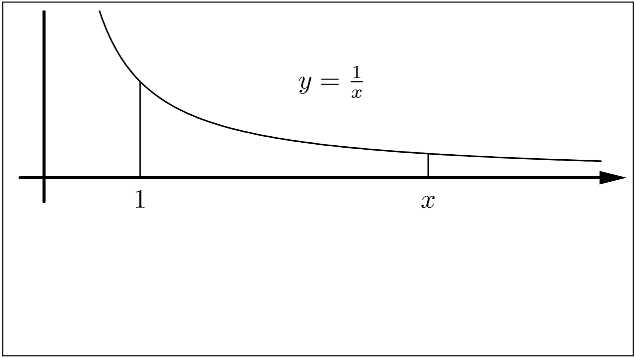
These numbers go to infinity:

$$\lim_{n \to \infty} H_n = \sum_{i=1}^{\infty} \frac{1}{i} = \infty$$

Harmonic Numbers

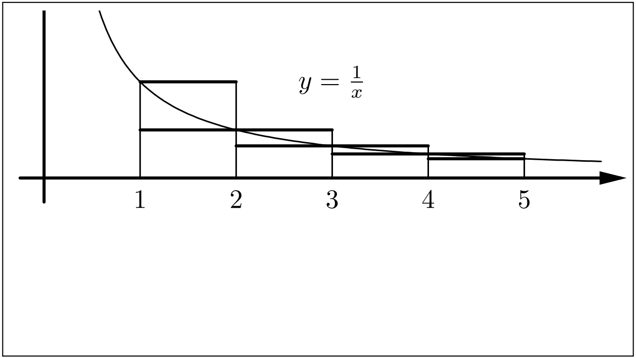
The harmonic numbers are closely related to logs. Recall:

$$\ln x = \int_1^x \frac{1}{t}\,dt$$



We will show that $H_n = \Theta(\log n)$.

Harmonic Numbers



$$\frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \quad < \quad \ln n \quad < \quad 1 + \frac{1}{2} + \ldots + \frac{1}{n-1}$$

$$H_n - 1 \quad < \quad \ln n \quad < \quad H_n - \frac{1}{n}$$

Hence $\ln n + \frac{1}{n} < H_n < \ln n + 1$, so $H_n = \Theta(\log n)$.

Proof/Justification Techniques

- Proof by Example Can be used to prove
  - A statement of the form "There exists…" is true.
  - A statement of the form "For all…" is false.
  - A statement of the form "If P then Q" is false.
- Illustration: Consider the statement:

  All numbers of the form $2^k - 1$ are prime.

  This statement is False: $2^4 - 1 = 15 = 3 \cdot 5$
- Note: The statement can be rewritten as:

  If n is an integer of the form $2^k - 1$, then n is prime.

Proof/Justification Techniques

- Suppose we want to prove a statement of the form "If P then Q" is true.
  There are three approaches:
  1. Direct proof: Assume P is true. Show that Q must be true.
  2. Indirect proof: Assume Q is false. Show that P must be false. This is also known as a proof by contraposition.
  3. Proof by contradiction: Assume P is true and Q is false. Show that there is a contradiction.

  See [GT] Section 1.3.3 for examples.

Proof/Justification Techniques: Induction

- A technique for proving theorems about the positive (or nonnegative) integers.
- Let $P(n)$ be a statement with an integer parameter, n. Mathematical induction is a technique for proving that $P(n)$ is true for all integers $\geq$ some base value b.
- Usually, the base value is 0 or 1.
- To show $P(n)$ holds for all $n \geq b$, we must show two things:
  1. Base Case: $P(b)$ is true (where b is the base value).
  2. Inductive step: If $P(k)$ is true, then $P(k+1)$ is true.

Induction Example

Example: Show that for all $n \geq 1$

$$\sum_{i=1}^{n} i \cdot 2^i = (n-1) \cdot 2^{(n+1)} + 2$$

Base Case: $(n = 1)$

$$\begin{aligned} \text{LHS} &= \sum_{i=1}^{1} i \cdot 2^i = 1 \cdot 2^1 = 2. \\ \text{RHS} &= (1-1) \cdot 2^{1+1} + 2 = 0 + 2 = 2. \\ \text{LHS} &= \text{RHS} \quad \checkmark \end{aligned}$$

Induction Example, continued

Inductive Step:

Assume $P(k)$ is true:

$$\sum_{i=1}^{k} i \cdot 2^i = (k-1) \cdot 2^{(k+1)} + 2.$$

Show $P(k+1)$ is true:

$$\sum_{i=1}^{k+1} i \cdot 2^i = k \cdot 2^{(k+2)} + 2.$$

Induction Example, continued

$$\text{Assume: } \sum_{i=1}^{k} i \cdot 2^i = (k-1) \cdot 2^{(k+1)} + 2.$$

$$\text{Show: } \sum_{i=1}^{k+1} i \cdot 2^i = k \cdot 2^{(k+2)} + 2.$$

$$\begin{aligned}
\sum_{i=1}^{k+1} i \cdot 2^i &= \sum_{i=1}^{k} i \cdot 2^i + (k+1) \cdot 2^{(k+1)} \\
&= (k-1) \cdot 2^{(k+1)} + 2 + (k+1) \cdot 2^{(k+1)} \\
&= 2k \cdot 2^{(k+1)} + 2 \\
&= k \cdot 2^{(k+2)} + 2
\end{aligned}$$

Probability

- Defined in terms of a sample space, S.
- Sample space consists of a finite set of outcomes, also called elementary events.
- An event is a subset of the sample space. (So an event is a set of outcomes).
- Sample space can be infinite, even uncountable. In this course, it will generally be finite.

Example: (2-coin example.) Flip two coins.

- Sample space $S = \{\mathtt{HH}, \mathtt{HT}, \mathtt{TH}, \mathtt{TT}\}$.
- The event "first coin is heads" is the subset $\{\mathtt{HH}, \mathtt{HT}\}$.

Probability function

- A probability function is a function $P(\cdot)$ that maps events (subsets of the sample space S) to real numbers such that:
  1. $P(\emptyset) = 0$.
  2. $P(S) = 1$.
  3. For every event A, $0 \leq P(A) \leq 1$.
  4. If $A, B \subseteq S$ and $A \cap B = \emptyset$, then $P(A \cup B) = P(A) + P(B)$.
- Note: Property 4 implies that if $A \subseteq B$ then $P(A) \leq P(B)$.

Probability function (continued)

For finite sample spaces, this can be simplified:

- Sample space $S = \{s_1, \ldots, s_k\}$,
- Each outcome S_i is assigned a probability $P(s_i)$, with

$$\sum_{i=1}^{k} P(s_i) = 1.$$

- The probability of an event $E \subseteq S$ is:

$$P(E) = \sum_{s_i \in E} P(s_i).$$

Example: (2-coin example, continued). Define

$$P(\mathtt{HH}) = P(\mathtt{HT}) = P(\mathtt{TH}) = P(\mathtt{TT}) = \frac{1}{4}.$$

Random variables

- Intuitive definition: a random variable is a variable whose value depends on the outcome of some experiment.
- Formal definition: a random variable is a function that maps outcomes in a sample space S to real numbers.
- Special case: An Indicator variable is a random variable that is always either 0 or 1.

Expectation

- The expected value, or expectation, of a random variable X represents its "average value".
- Formally: Let X be a random variable with a finite set of possible values $\mathrm{V} = \{x_1, \ldots, x_k\}$. Then

$$E(X) = \sum_{x \in V} x \cdot P(X = x).$$

Example: (2-coin example, continued). Let X be the number of heads when two coins are thrown. Then

$$\begin{aligned} E(X) &= 0 \cdot P(X=0) + 1 \cdot P(X=1) + 2 \cdot P(X=2) \\ &= 0 \cdot \left(\frac{1}{4}\right) + 1 \cdot \left(\frac{1}{2}\right) + 2 \cdot \left(\frac{1}{4}\right) \\ &= 1 \end{aligned}$$

Expectation

Example: Throw a single six-sided die. Assume the die is fair, so each possible throw has a probability of $1/6$.

The expected value of the throw is:

$$1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} = 3.5$$

Linearity of Expectation

- For any two random variables X and Y,

$$E(X+Y) = E(X) + E(Y).$$

- Proof: see [GT], 1.3.4
- Very useful, because usually it is easier to compute $E(X)$ and $E(Y)$ and apply the formula than to compute $E(X+Y)$ directly.

Example 1: Throw two six-sided dice. Let X be the sum of the values. Then

$$E(X) = E(X_1 + X_2) = E(X_1) + E(X_2) = 3.5 + 3.5 = 7,$$

where X_i is the value on die i ($i = 1, 2$).

Example 2: Throw 100 six-sided dice. Let Y be the sum of the values. Then

$$E(Y) = 100 \cdot 3.5 = 350.$$

Independent events

- Two events A_1 and A_2 are independent iff

$$P(A_1 \cap A_2) = P(A_1) \cdot P(A_2).$$

Example: (2-coin example, continued). Let

$$\begin{array}{lcl} A_1 & = & \text{coin 1 is heads} & = & \{\mathtt{HH}, \mathtt{HT}\} \\ A_2 & = & \text{coin 2 is tails} & = & \{\mathtt{HT}, \mathtt{TT}\} \end{array}$$

Then $P(A_1) = \frac{1}{2}$, $P(A_2) = \frac{1}{2}$, and

$$P(A_1 \cap A_2) = P(\mathtt{HT}) = \frac{1}{4} = P(A_1) \cdot P(A_2).$$

So A_1 and A_2 are independent.

Independent events

A collection of n events $C = \{A_1, A_2, \ldots, A_n\}$ is mutually independent (or simply independent) if:

For every subset $\{A_{i_1}, A_{i_2}, \ldots A_{i_k}\} \subseteq C$:

$$P(A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_k}) = P(A_{i_1}) \cdot P(A_{i_2}) \cdots P(A_{i_k}).$$

Example: Suppose we flip 10 coins. Suppose the flips are fair ($P(\mathtt{H}) = P(\mathtt{T}) = 1/2$) and independent. Then the probability of any particular sequence of flips (e.g., `HHTTTHTHTH`) is $1/(2^{10})$.

Example: Probability and counting

Example: Suppose we flip a coin 10 times. Suppose the flips are fair and independent. What is the probability of getting exactly 7 heads out of the 10 flips?

Solution:

- The outcomes consist of the set of possible sequences of 10 flips (e.g., `HHTTTHTHTH`).
- The probability of each outcome is $1/(2^{10})$.
- The number of successful outcomes is $\binom{10}{7}$.
- Hence the probability of getting exactly 7 heads is:

$$\frac{\binom{10}{7}}{2^{10}} = \frac{120}{1024} = 0.117.$$

An average-case result about finding the maximum

```
v = -∞
for i = 0 to n-1:
   if A[i] > v:
      v = A[i]
return v
```

- Worst-case number of comparisons is n.
- This can be reduced to $n - 1$
- How many times is the running maximum updated?
  - In the worst case n.
  - What about the average case? ...

Average number of updates to the running maximum

- Assume
  - all possible orderings (permutations) of A are equally likely
  - all n elements of A are distinct.
- The running maximum gets updated on iteration i of the loop iff $\max\{A[0], \ldots, A[i]\} = A[i]$.
- The probability of this happening is $1/(i+1)$.
- Define indicator variables X_i:

$$X_i = \begin{cases} 1 & \text{if } v \text{ gets updated on iteration } \#i \\ 0 & \text{if } v \text{ does not get updated on iteration } \#i \end{cases}$$

  Then $E(X_i) = \frac{1}{i+1}$
- The total number of times that v gets updated is:

$$X = \sum_{i=0}^{n-1} X_i$$

Average number of updates to the running maximum (continued)

The expected total number of times that v gets updated is:

$$E(X) = E\left(\sum_{i=0}^{n-1} X_i\right) = \sum_{i=0}^{n-1} E\left(X_i\right) = \sum_{i=0}^{n-1} \frac{1}{i+1} = \sum_{i=1}^{n} \frac{1}{i} = H_n = O(\log n)$$

It can be shown that

$$H_n = \ln n + \gamma + o(1), \quad \text{where } \gamma = 0.5772157\ldots$$

If there are 30,000 elements in the list, the expected update count is about 10.9

If there are 3,000,000,000 elements in the list, the expected update count is about 22.4