Introduction and Preliminaries

Overview

- Introduction to analysis of algorithms
- Asymptotic notation
- Review of Mathematical Background

Books

Text book:

- [GT] Goodrich and Tamassia, *Algorithm design and Applications*, Wiley.

A few other books worth knowing about:

- [BvG] Baase and van Gelder, *Computer Algorithms*, Addison-Wesley.
- [KT] Kleinberg and Tardos, *Algorithm Design*, Addison Wesley.
- [CLRS] Cormen, Leiserson, Rivest, and Stein, *Introduction to Algorithms*, MIT Press.

Algorithms

What is an algorithm?

- The classical definition is summarized in the text [GT, Section 1.1]:

  > *Simply put, an* **algorithm** *is a step-by-step procedure for performing some task in a finite amount of time ...*

- In popular usage, the word "algorithms" is sometimes used to mean "things done by a computer," particularly when predictive or prescriptive analytics is involved.
- In this course we will use the classical definition.

This course will focus on the analysis and design of algorithms, as defined above.

Analysis of Algorithms

Why bother analyzing algorithms? Why not just implement them and run them?

- Predict behavior before implementation.
- Helps choose among different solutions.
- Experimentation is limited by the test cases on which experimentation is performed.
- Experimentation may be biased by choice of hardware.
- Provides insight into possible improvements.

Analysis of Algorithms

What do we analyze?

- Correctness
  - It is very easy to develop fast, incorrect algorithms.
  - However, this is not very useful.
- Performance
  - Qualitative: work, storage, disk accesses, communication, ...?
    - Most commonly: running time.
  - Quantitative: what do we count?
    - Need a methodology, quantifiable model.
- Scalability: What happens as our inputs get large?

Analysis of Algorithms

We need...

- Methodology for algorithm analysis
  - Way of describing algorithms
  - A computational model
  - A metric for measuring running times
  - An approach for characterizing running times
- Mathematical tools for performing the analysis and expressing the results
  - Terminology for expressing scalability (Asymptotic notation)
  - Some basic functions
  - Techniques for reasoning, justification
  - Probability

Describing Algorithms: Pseudocode

- Intended for humans, not machines
- Not computer programs, but more structured than natural language
- Focus on high-level ideas, not low-level implementation details
- Captures important steps
- [GT] enumerate some reasonable constructs
- Requires finding the right balance

On the next few slides, we will give some examples of algorithms written using the pseudocode constructs described in the textbook.

Note:

- Most of the examples presented later on in these notes will use pseudocode based on Python.

Example 1: Finding Maximum, version 1

Algorithm Maximum1(A,n)
Input: An array $A[n]$, where $n \geq 1$
Output: A maximum element in A

$v \leftarrow A[0]$
for $i \leftarrow 1$ to $n-1$ do
    if $A[i] > v$ then
        $v \leftarrow A[i]$
return v

- Correctness: At start of iteration i, v is “leftmost maximum” of first i array entries.

Example 2: Finding Maximum, version 2

Algorithm Maximum2(A,n)
Input: An array $A[n]$, where $n \geq 1$
Output: A maximum element in A

$v \leftarrow -\infty$
for $i \leftarrow 0$ to $n-1$ do
    if $A[i] \geq v$ then
        $v \leftarrow A[i]$
return v

- Correctness: At start of iteration i, v is "rightmost maximum" of first i array entries.
- What is the maximum of an empty collection?

Example 3: Sequential Search

Determine whether an array contains a particular item and, if so, an return an index at which the item is stored. If the item is not stored in the array, return the special value -1.

Algorithm Search(A,n,x)
Input: An array $A[n]$, where $n \geq 1$; an item x
Output: Index where x occurs in A, or -1

for $i \leftarrow 0$ to $n-1$ do
    if $A[i] = x$ then return(i)
return(-1)

- Correctness: At start of iteration i, either we have returned a correct value or x is not one of the first i entries.

Computational Model: RAM

Random Access Machine (RAM)

- Define primitive operations
- Operations include:
  - Assigning a value to a variable
  - Calling a function (method)
  - Performing an arithmetic operation
  - Comparing two numbers
  - Indexing into an array
  - Following an object reference (Dereferencing a pointer)
  - Returning from a function
  - Transfer of control (jump/goto)
- To measure running time: count operations

Example: Finding Maximum

$v \leftarrow A[0]$	2
// for $i \leftarrow 1$ to $n-1$ do	
$i \leftarrow 1$	1
Loop:	
if $i \leq n-1$ then	$2n$
if $A[i] > v$ then	$2n-2$
$v \leftarrow A[i]$	between 0 and $2n-2$,inclusive
$i = i + 1$	$2n-2$
go to Loop	$n-1$
return v	1

- Best case: cost $= 7n - 1$
- Worst case: cost $= 9n - 3$

Best case vs. Worst case vs. Average Case

- Algorithms run faster on some inputs than others
- What about average case (taken over all inputs)?
  - Often requires heavy mathematics, probability
  - Requires knowing the probability distribution on the set of inputs. This can be hard to determine.
- We will focus on worst-case analysis
  - Gives us a guarantee.
  - Murphy's law: "If anything can go wrong, it will."
  - If we design for worst case, sometimes we get a better algorithm
- Another type of average case analysis: algorithm makes random decisions.

Recursive Algorithms
Alternative to Iterative Algorithms

Example: Finding maximum

recursiveMax(A,n) // returns maximum of A[0]...A[n-1]	
if $n = 1$ then	1
return A[0]	2
return max(recursiveMax(A,n-1),A[n-1])	$T(n-1)+6$

- Let $T(n)$ be cost for input of size n. Equation:

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

- Solution: $T(n) = 7n - 4$

Asymptotic notation: O,o,Ω,Θ

- Measures growth rates of functions as $n \to \infty$.
- Treats two functions as being roughly the same if they are roughly constant multiples of each other.
- Function may represent description of algorithm behavior. (E.g., $f(n)$ could be the worst-case running time of a given algorithm on an input of size n).
- Allows us to focus on the most important considerations when analyzing an algorithm, and to ignore fine-grain details.
- Allows us to compare two algorithms easily.
- (Allows us to be "imprecise in a precise way")

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:

$$\exists_{C>0}\, \exists_{n_0 \geq 1}\, \forall_{n \geq n_0}\, g(n) \leq C \cdot f(n).$$

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

$$\exists_{C>0}\, \exists_{n_0 \geq 1}\, \forall_{n \geq n_0}\, \frac{g(n)}{f(n)} \leq C.$$

- 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 positive n.

So we can choose $C = 1000$ and $n_0 = 1$.

Examples of O-notation:

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

Proof: $\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0$.

Hence for any $C > 0$ the ratio is less than C when 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 $n^{3/2} \leq n^2$.

So 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: $\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{n^4}{n^3} = \lim_{n\to\infty} n = \infty$.

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")

$$g \in \Omega(f) \quad \text{if and only if} \quad \exists_{C>0}\, \exists_{n_0 \geq 1}\, \forall_{n \geq n_0}\; g(n) \geq C \cdot f(n).$$

- Equivalently:

$$g \in \Omega(f) \quad \text{if and only if} \quad \exists_{C>0}\, \exists_{n_0 \geq 1}\, \forall_{n \geq n_0}\; \frac{g(n)}{f(n)} \geq C.$$

- Ω and O are inverses of each other:

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

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:

$$\exists_{C_1>0}\, \exists_{C_2>0}\, \exists_{n_0\geq 1}\, \forall_{n\geq n_0}\; C_1 \leq \frac{g(n)}{f(n)} \leq C_2.$$

- If $\lim_{n\to\infty} \frac{g(n)}{f(n)}$ exists and is not zero, then $g \in \Theta(f)$.
  - The converse is not necessarily true.

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

We will show that $g \in \Omega(f)$.

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

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 $C = 4$, $n_0 = 23$.

Note: $g \in O(f)$ (Exercise: show this). Hence $g \in \Theta(f)$.

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

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

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

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

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 $g = o(f)$.

Final Note on Aymptotic Notation

Read material on asymptotic notation in the [GT] (the second half of Section 1.2,immediatly after Activity 1.2.5). Note in particular:

- Theorem 1.2.2 summarizes some very important and useful facts and properties. Here are two of them:
  6. n^x is $O(a^n)$ for any fixed $x > 0$ and $a > 1$.
  8. $(\log n)^x$ is $O(n^y)$ for any fixed constants $x > 0$ and $y > 0$.
- There are some additional examples beyond those presented here.
- The subsection entitled "Some Words of Caution" [begins after Example 1.2.8]
- Specifics about functions and running time (towards the end of the section).

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. $\log_a x = (\log_b x)(\log_a b)$.

Exercise: Prove the above properties.

Logarithms

Example (#2): Prove $\log_b b^a = a$.

Solution: Let
$$\begin{aligned} y &= \log_b b^a \\ b^y &= b^a \quad \text{[by definition of log]} \\ y &= a \end{aligned}$$

Logarithms

Special Notations:

- $\ln x = \log_e x$ ($e = 2.71828\ldots$)
- $\lg x = \log_2 x$
  - This notation is used in the notes. It is also used in [CLRS].
  - [GT] uses $\log x$ without a base to represent $\log_2 x$. This is explicitly stated in the text in Section 1.3, at the start of the subsection on logarithms and exponents

Some conversions (from Rules #7 and #8 on previous slides):

- $\ln x = (\log_2 x)(\log_e 2) = 0.693 \lg x$.
- $\lg x = \frac{\log_e x}{\log_e 2} = \frac{\ln x}{0.693} = 1.44 \ln x$.

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)

Examples:

- $\lfloor 3.5 \rfloor = 3$
- $\lceil 3.5 \rceil = 4$
- $\lfloor 3 \rfloor = \lceil 3 \rceil = 3$
- $\lfloor -3.5 \rfloor = -4$
- $\lceil -3.5 \rceil = -3$
- $\lfloor -3 \rfloor = \lceil -3 \rceil = -3$

Rule for negating floors and ceilings:

- $\lfloor -x \rfloor = -\lceil x \rceil$
- $\lceil -x \rceil = -\lfloor x \rfloor$

Factorials

- $n!$, read as "n factorial," is defined for integers $n \geq 0$
- $n! = 1 \cdot 2 \cdots n$
- Equivalently:

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{if } n \geq 1 \end{cases}$$

- $n!$ 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 a subset of k objects from a set 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: $\binom{n}{0} = 1$, $\binom{n}{1} = n$, $\binom{n}{2} = \frac{n(n-1)}{2} \in \Theta\left(n^2\right)$

Harmonic Numbers

The *n*th Harmonic number is the sum:

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

These numbers go to infinity:

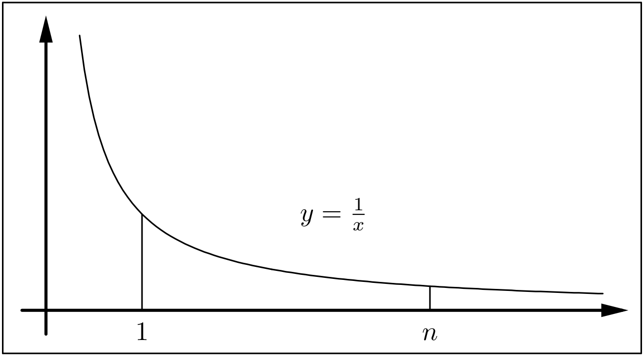
$$\lim_{n\to\infty} H_n = \sum_{i=1}^{\infty} \frac{1}{i} = \infty$$

Harmonic Numbers

The harmonic numbers are closely related to logs.

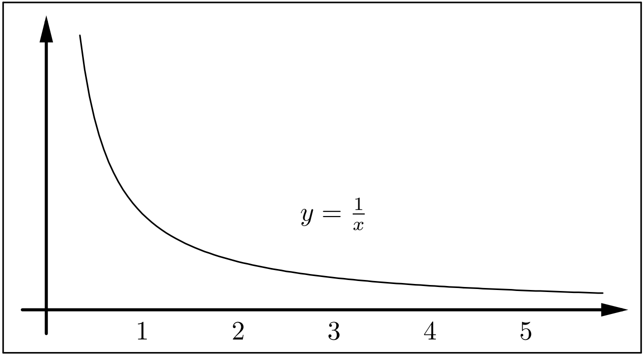
Recall from calculus:

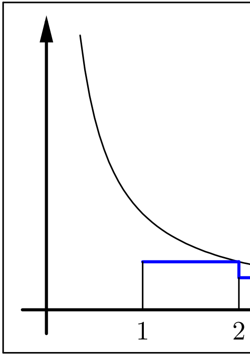
$$\ln n = \int_1^n \frac{1}{x}\, dx$$



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 positive integers 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 a positive 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. Proof by contraposition: Assume Q is false. Show that P must be false.
    3. Proof by contradiction: Assume P is true and Q is false. Show that there is a contradiction.
- Note: Some textbooks refer to proof by contraposition and/or proof by contradiction as Indirect proofs.
- A few examples are in [GT] Section 1.3.

Proofs by contradiction are often the easiest to construct but the hardest to understand.

Proof/Justification Techniques: Induction

Induction: (sometimes called mathematial induction):

- A technique for proving theorems about the positive (or nonnegative) integers.
- Let $P(n)$ be a statement with an integer parameter, n. 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 for a value $k \geq b$, then $P(k+1)$ is true.

  The Principle of Mathematical Induction says that if the base case and the inductive step hold, then the statement $P(n)$ is true for all $n \geq b$.

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

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

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

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

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).
- In general sample space can be infinite, even uncountable. In this course, our sample spaces will always 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 models how likely it is that we we run an experiment, a given event will occur.

- 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 $P(s_i)$, where
  1. $P(s_i) \geq 0$ for all i
  2. $\displaystyle\sum_{i=1}^{k} P(s_i) = 1$
- The probability of an event $E \subseteq S$ is:

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

Example: (2-coin example, continued). $S = \{\mathtt{HH}, \mathtt{HT}, \mathtt{TH}, \mathtt{TT}\}$.

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

$$\text{Then } P(\text{first coin is heads}) = P(\mathtt{HH}) + P(\mathtt{HT}) = \frac{1}{2}.$$

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.

Expected Value

- The expected value, or expectation, of a random variable X represents its "average value".
  - It is denoted by $E(X)$.
  - Sometimes it is just written as EX.
- Formally: Let X be a random variable with a finite set of possible values $V = \{x_1, \ldots, x_k\}$. Then

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

Example: (2-coin example, continued). $S = \{\texttt{HH}, \texttt{HT}, \texttt{TH}, \texttt{TT}\}$, and the probability of each outcome is $1/4$.

Let X be the number of heads when two coins are thrown. Compute $E(X)$

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

Expected Value

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,

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

- 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.
- Proof: [GT], (Theorem 1.3.5)

Example 1: Throw two six-sided dice. Let X be the sum of the values. Let X_i be the value on die i ($i = 1, 2$). Then

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

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

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

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

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

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`). There are 2^{10} such sequences.
- 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]v = A[i]
return v
```

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

Average number of updates to the candidate maximum

- Assume
  - all possible orderings (permutations) of A are equally likely
  - all n elements of A are distinct.
- The candidate 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 candidate maximum (continued)

The expected total number of times that v gets updated is $E(X)$.

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

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