Algorithms

University of Virginia

Gabriel Robins
Course Outline

• Historical perspectives
• Foundations
• Data structures
• Sorting
• Graph algorithms
• Geometric algorithms
• Statistical analysis
• NP-completeness
• Approximation algorithms
Prerequisites

Some discrete math / algorithms knowledge would be helpful (but is not necessary)

Textbook

Suggested Reading


“This book fills a much-needed gap.”
- Moses Hadas (1900-1966) in a review
Grading scheme

Midterm: 35%

Final: 35%

Project: 30%

Extra credit: 10%

“The mistakes are all there waiting to be made.”
- chessmaster Savielly Grigorievitch Tartakower (1887-1956)
on the game’s opening position
Specifics

• Homeworks

• Solutions

• Extra-credit
  • In-class
  • Find mistakes

• Office hours: after class
  • Any time
  • Email (preferred)
  • By appointment
  • Q&A posted on the Web

• Exams: take home?
Contact Information

Prof: Gabriel Robins
Office: 406 Rice Hall
Phone: (434) 982-2207
EMail: robins@cs.virginia.edu

www.cs.virginia.edu/~robins

“Good teaching is one-fourth preparation and three-fourths theater.” - Gail Godwin
Good Advice

• Ask questions ASAP
• Do homeworks ASAP
• Do not fall behind
• “Cramming” won’t work
• Start on project early
• Attend every lecture
• Read Email often
• Solve lots of problems
Basic Questions/Goals

Q: How do you solve problems?
   • Proof techniques

Q: What resources are needed to compute certain functions?
   • Time / space / “hardware”

Q: What makes problems hard/easy?
   • Problem classification

Q: What are the fundamental limitations of algorithms?
   • Computability / undecidability
Historical Perspectives

• Euclid (325BC – 265BC)
  “Elements”

• Rene Descartes (1596-1650)
  Cartesian coordinates

• Pierre de Fermat (1601-1665)
  Fermat’s Last Theorem

• Blaise Pascal (1623-1662)
  Probability

• Leonhard Euler (1707-1783)
  Graph theory
• Carl Friedrich Gauss (1777-1855)  
Number theory

• George Boole (1815-1864)  
Boolean algebra

• Augustus De Morgan (1806-1871)  
Symbolic logic, induction

• Ada Augusta (1815-1852)  
Babbage’s Analytic Engine

• Charles Dodgson (1832-1898)  
Alice in Wonderland

• John Venn (1834-1923)  
Set theory and logic
• Georg Cantor (1845-1918)  
  Transfinite arithmetic

• Bertrand Russell (1872-1970)  
  “Principia Mathematica”

• Kurt Godel (1906-1978)  
  Incompleteness

• Alan Turing (1912-1954)  
  Computability

• Alonzo Church (1903-1995)  
  Lambda-calculus

• John von Neummann (1903-1957)  
  Stored program
• Claude Shannon (1916-2001)  
  Information theory

• Stephen Kleene (1909-1994)  
  Recursive functions

• Noam Chomsky (1928-)  
  Formal languages

• John Backus (1924-)  
  Functional programming

• Edsger Dijkstra (1930-2002)  
  Structured programming

• Paul Erdos (1913-1996)  
  Combinatorics
Symbolic Logic

Def: proposition - statement either true (T) or false (F)

Ex: 1+1=2

2+2=3

“today is Monday”

“what time is it?”

x + 4 = 5
Boolean Functions

- “and” \( \land \)
- “or” \( \lor \)
- “not” \( \neg \)
- “xor” \( \oplus \)
- “nand”
- “nor”
- “implication” \( \Rightarrow \)
- “equivalence” \( \Leftrightarrow \)
• “not” \( \neg \) “negation”

Truth table:

<table>
<thead>
<tr>
<th>( p )</th>
<th>( \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Ex: let \( p = \)“today is Monday”

\( \neg p = \)“today is not Monday”
• “and” $\land$

“conjunction”

Truth table:

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>$p \land q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Ex: $x \geq 0 \land x \leq 10$

$(x \geq 0) \land (x \leq 10)$
• “or” \( \lor \)  

“disjunction”

Truth table:

\[
\begin{array}{ccc}
 p & q & p \lor q \\
 T & T & T \\
 T & F & T \\
 F & T & T \\
 F & F & F \\
\end{array}
\]

Ex:  \((x \geq 7) \lor (x = 3)\) 

\((x = 0) \lor (y = 0)\)
• “xor” \( \oplus \) “exclusive or”

Truth table:

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>( p \oplus q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Ex: \((x=0) \oplus (y=0)\)

“it is midnight” \( \oplus \) “it is sunny”
Logical Implication

• “implies” \( \Rightarrow \)

Truth table:

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p \implies q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Ex: \((x \leq 0) \land (x \geq 0) \Rightarrow (x=0)\)

\(1 < x < y \Rightarrow x^3 < y^3\)

“today is Sunday” \(\Rightarrow 1+1=3\)
Other interpretations of $p \implies q$:

- “$p$ implies $q$”
- “if $p$, then $q$”
- “$q$ only if $p$”
- “$p$ is sufficient for $q$”
- “$q$ if $p$”
- “$q$ whenever $p$”
- “$q$ is necessary for $p$”
Logical Equivalence

• “biconditional” $\iff$
  
or “if and only if” (“iff”)
  
or “necessary and sufficient”
  
or “logically equivalent” $\equiv$

Truth table:

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>$p \iff q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Ex: $p \iff p$

$[(x=0) \lor (y=0)] \iff (xy=0)$

$\min(x,y)=\max(x,y) \iff x=y$
logically equivalent (⇔) - means “has same truth table”

Ex: \( p \implies q \) is equivalent to \( (\neg p) \lor q \)

i.e., \( p \implies q \iff (\neg p) \lor q \)

\[
\begin{array}{|c|c|c|c|c|}
\hline
p & q & p \implies q & \neg p & \neg p \lor q \\
\hline
T & T & T & F & T \\
T & F & F & F & F \\
F & T & T & T & T \\
F & F & T & T & T \\
\hline
\end{array}
\]

Ex: \( (p \iff q) \equiv [(p \implies q) \land (q \implies p)] \)

\( p \iff q \equiv p \implies q \land q \implies p \)

\( (p \iff q) \equiv [(\neg p \lor q) \land (\neg q \lor p)] \)
Note: $p \implies q$ is not equivalent to $q \implies p$

Thm: $(P \implies Q) \equiv (\neg Q \implies \neg P)$

Q: What is the negation of $p \implies q$?

A: $\neg(p \implies q) \equiv \neg(\neg p \lor q) \equiv p \land \neg q$

| $p$ | $q$ | $\neg q$ | $p \implies q$ | $\neg(p \implies q)$ | $p \land \neg q$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

“Logic is in the eye of the logician.”
- Gloria Steinem
Example

let p = “it is raining”
let q = “the ground is wet”

$p \Rightarrow q : \text{“if it is raining, then the ground is wet”}$

$q \Rightarrow \neg p : \text{“if the ground is not wet, then it is not raining”}$

$q \Rightarrow p : \text{“if the ground is wet, then it is raining”}$

$\neg(p \Rightarrow q) : \text{“it is raining, and the ground is not wet”}$
Order of Operations

• negation first
• or/and next
• implications last
• parenthesis override others

(similar to arithmetic)

Def: converse of $p \implies q$ is $q \implies p$

contrapositive of $p \implies q$ is $\neg q \implies \neg p$

Prove: $p \implies q \equiv \neg q \implies \neg p$
Q: How many distinct 2-variable Boolean functions are there?
Bit Operations

\[ \neg \]

\[
\begin{array}{c|c}
0 & 1 \\
1 & 0 \\
\end{array}
\]

\[ \land \]

\[
\begin{array}{c|c|c}
 & 0 & 1 \\
0 & 0 & 0 \\
1 & 0 & 1 \\
\end{array}
\]

\[ \lor \]

\[
\begin{array}{c|c|c}
 & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

\[ \Rightarrow \]

\[
\begin{array}{c|c|c}
 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
\end{array}
\]

\[ \Leftrightarrow \]

\[
\begin{array}{c|c|c}
 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
\end{array}
\]
Bit Strings

Def: *bit string* - sequence of bits

Boolean functions extend to bit strings (bitwise)

Ex: \( \overline{0100} = 1011 \)

\[ 0100 \land 1110 = 0100 \]
\[ 0100 \lor 1110 = 1110 \]
\[ 0100 \oplus 1110 = 1010 \]
\[ 0100 \Rightarrow 1110 = 1111 \]
\[ 0100 \Leftrightarrow 1110 = 0101 \]
Proposition types

Def: *tautology*: always true
*contingency*: sometimes true
*contradiction*: never true

Ex: \( p \lor \neg p \) is a tautology
\( p \land \neg p \) is a contradiction
\( p \Rightarrow \neg p \) is a contingency

<table>
<thead>
<tr>
<th></th>
<th>( p )</th>
<th>( \neg p )</th>
<th>( p \lor \neg p )</th>
<th>( p \land \neg p )</th>
<th>( p \Rightarrow \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Logic Laws

Identity:

\[ p \land T \iff p \]
\[ p \lor F \iff p \]

Domination:

\[ p \lor T \iff T \]
\[ p \land F \iff F \]

Idempotent:

\[ p \lor p \iff p \]
\[ p \land p \iff p \]
Logic Laws (cont.)

Double Negation:

$$\neg(\neg p) \iff p$$

Commutative:

$$p \lor q \iff q \lor p$$
$$p \land q \iff q \land p$$

Associative:

$$(p \lor q) \lor r \iff p \lor (q \lor r)$$
$$(p \land q) \land r \iff p \land (q \land r)$$
Distributive:

\[ p \lor (q \land r) \iff (p \lor q) \land (p \lor r) \]

\[ p \land (q \lor r) \iff (p \land q) \lor (p \land r) \]

De Morgan’s:

\[ \neg (p \lor q) \iff \neg p \land \neg q \]

\[ \neg (p \land q) \iff \neg p \lor \neg q \]

Misc:

\[ p \lor \neg p \iff T \]

\[ p \land \neg p \iff F \]

\[ (p \Rightarrow q) \iff (\neg p \lor q) \]
Example

Simplify the following:

$$(p \land q) \Rightarrow (p \lor q)$$
Def: *predicate* - a function or formula involving some variables

Ex: let $P(x) = \text{"}x > 3\text{"}$
    $x$ is the variable
    "$x>3$" is the predicate

$P(5)$

$P(1)$

Ex: $Q(x,y,z) = \text{"} x^2+y^2=z^2 \text{"}$

$Q(2,3,4)$

$Q(3,4,5)$
Quantifiers

- Universal: “for all”  \( \forall \)
  \[ \forall x \ P(x) \]
  \[ \iff P(x_1) \land P(x_2) \land P(x_3) \land \ldots \]
  Ex:
  \[ \forall x \ x < x + 1 \]
  \[ \forall x \ x < x^3 \]

- Existential: “there exists”  \( \exists \)
  \[ \exists x \ P(x) \]
  \[ \iff P(x_1) \lor P(x_2) \lor P(x_3) \lor \ldots \]
  Ex:
  \[ \exists x \ x = x^2 \]
  \[ \exists x \ x < x - 1 \]

Combinations:
\[ \forall x \exists y \ y > x \]
Examples

• $\forall x \exists y \ x+y=0$

• $\exists y \forall x \ x+y=0$

• “every dog has his day”:
  $\forall d \exists y \ H(d,y)$

• $\lim_{x \to a} f(x) = L$

  $\forall \varepsilon \exists \delta \forall x \ (0 < |x-a| < \delta \Rightarrow |f(x)-L| < \varepsilon)$
Examples (cont.)

• n is divisible by j (denoted n|j):
  \[ n|j \iff \exists k \in \mathbb{Z} \ n = kj \]

• m is prime (denoted P(m)):
  \[ P(m) \iff [\forall i \in \mathbb{Z} \ (m|i) \Rightarrow (i=m) \lor (i=1)] \]

• “there is no largest prime”
  \[ \forall p \ \exists q \in \mathbb{Z} \ (q>p) \land P(q) \]
  \[ \forall p \ \exists q \in \mathbb{Z} \ (q>p) \land [\forall i \in \mathbb{Z} \ (q|i) \Rightarrow (i=q) \lor (i=1)] \]
  \[ \forall p \ \exists q \in \mathbb{Z} \ (q>p) \land [\forall i \in \mathbb{Z} \ \{\exists k \in \mathbb{Z} \ q = ki\} \Rightarrow (i=q) \lor (i=1)] \]
Negation of Quantifiers

Thm: \( \neg (\forall x \ P(x)) \iff \exists x \ \neg P(x) \)

Ex: \( \neg "\text{all men are mortal}" \iff "\text{there is a man who is not mortal}" \)

Thm: \( \neg (\exists x \ P(x)) \iff \forall x \ \neg P(x) \)

Ex: \( \neg "\text{there is a planet with life on it}" \iff "\text{all planets do not contain life}" \)

Thm: \( \neg \exists x \forall y \ P(x,y) \iff \forall x \exists y \ \neg P(x,y) \)

Ex: \( \neg "\text{there is a man that exercises every day}" \iff "\text{every man does not exercise some day}" \)

Thm: \( \neg \forall x \exists y \ P(x,y) \iff \exists x \forall y \ \neg P(x,y) \)

Ex: \( \neg "\text{all things come to an end}" \iff "\text{some thing does not come to any end}" \)
Quantification Laws

Thm: $\forall x (P(x) \land Q(x))$
  $\iff (\forall x P(x)) \land (\forall x Q(x))$

Thm: $\exists x (P(x) \lor Q(x))$
  $\iff (\exists x P(x)) \lor (\exists x Q(x))$

Q: Are the following true?

$\exists x (P(x) \land Q(x))$
  $\iff (\exists x P(x)) \land (\exists x Q(x))$

$\forall x (P(x) \lor Q(x))$
  $\iff (\forall x P(x)) \lor (\forall x Q(x))$
More Quantification Laws

- \((\forall x \ Q(x)) \land P \iff \forall x \ (Q(x) \land P)\)

- \((\exists x \ Q(x)) \land P \iff \exists x \ (Q(x) \land P)\)

- \((\forall x \ Q(x)) \lor P \iff \forall x \ (Q(x) \lor P)\)

- \((\exists x \ Q(x)) \lor P \iff \exists x \ (Q(x) \lor P)\)
Unique Existence

Def: \( \exists! x \ P(x) \) means there exists a unique \( x \) such that \( P(x) \) holds

Q: Express \( \exists! x \ P(x) \) in terms of the other logic operators

A:
Mathematical Statements

- Definition
- Lemma
- Theorem
- Corollary

Proof Types

- Construction
- Contradiction
- Induction
- Counter-example
- Existence
- …
Sets

Def: *set* - an unordered collection of elements

Ex: \{1, 2, 3\} or \{hi, there\}

Venn Diagram:

\[ S \]

Def: two sets are *equal* iff they contain the same elements

Ex: \{1, 2, 3\} = \{2, 3, 1\}

\{0\} \neq \{1\}

\{3, 5\} = \{3, 5, 3, 3, 5\}
• Set construction: or \( \exists \) means “such that”

Ex: \( \{k \mid 0 < k < 4\} \)

\( \{k \mid k \text{ is a perfect square}\} \)

• Set membership: \( \in \) \( \notin \)

Ex: \( 7 \in \{p \mid p \text{ prime}\} \)

\( q \notin \{0, 2, 4, 6, \ldots\} \)

• Sets can contain other sets

Ex: \( \{2, \{5\}\} \)

\( \left\{\left\{\left\{0\right\}\right\}\right\} \neq \{0\} \neq 0 \)

\( S = \{1, 2, 3, \{1\}, \{\{2\}\}\} \)
Common Sets

Naturals: \( \mathbb{N} = \{1, 2, 3, 4, \ldots\} \)

Integers: \( \mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\} \)

Rationals: \( \mathbb{Q} = \left\{ \frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0 \right\} \)

Reals: \( \mathbb{R} = \{x \mid x \text{ a real } \#\} \)

Empty set: \( \emptyset = \{\} \)

\( \mathbb{Z}^+ \) = non-negative integers

\( \mathbb{R}^- \) = non-positive reals, etc.
Multisets

Def: a set w/repeated elements allowed (i.e., each element has “multiplier”)

Ex: \{0, 1, 2, 2, 2, 5, 5\}

For multisets: \{3, 5\} \neq \{3, 5, 3, 3, 5\}

Sequences

Def: ordered list of elements

Ex: (0, 1, 2, 5) “4-tuple”
(1,2) \neq (2,1) “2-tuple”
Subsets

- **Subset notation:**
  \[ S \subseteq T \iff (x \in S \Rightarrow x \in T) \]

- **Proper subset:**
  \[ S \subset T \iff ((S \subseteq T) \land (S \neq T)) \]
  \[ S = T \iff ((T \subseteq S) \land (S \subseteq T)) \]
  \[ \forall S \quad \emptyset \subseteq S \]
  \[ \forall S \quad S \subseteq S \]
• **Union:**

\[ S \cup T = \{ x \mid x \in S \lor x \in T \} \]

• **Intersection:**

\[ S \cap T = \{ x \mid x \in S \land x \in T \} \]
• **Set difference**: \( S - T \)

\[
S - T = \{ x \mid x \in S \land x \not\in T \}
\]

\[\text{Diagram: } S \cap T\]

• **Symmetric difference**: \( S \oplus T \)

\[
S \oplus T = \{ x \mid x \in S \oplus x \in T \}
\]

\[
= S \cup T - S \cap T
\]

\[\text{Diagram: } S \cup T\]
• Universal set: \( U \) (everything)

• Set complement: \( S' \) or \( \overline{S} \)

\[
S' = \{ x \mid x \notin S \} = U - S
\]

• Disjoint sets: \( S \cap T = \emptyset \)

\[
S - T = S \cap T'
\]

\[
S - S = \emptyset
\]
Examples

\[ N \cup \mathbb{Z} \cup \mathbb{Q} \cup \mathbb{R} = \mathbb{R} \]

\[ N \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \]

\[ \forall x \in \mathbb{R} \quad x \leq x^2 + 1 \]

\[ \forall x, y \in \mathbb{Q} \quad \min(x, y) = \max(x, y) \iff x = y \]

\[ \mathbb{R}^+ \cup \mathbb{R}^- = \mathbb{R} \]

\[ \mathbb{R}^+ \cap \mathbb{R}^- = \{0\} \]
Set Identities

• **Identity:**
  \[ S \cup \emptyset = S \]
  \[ S \cap U = S \]

• **Domination:**
  \[ S \cup U = U \]
  \[ S \cap \emptyset = \emptyset \]

• **Idempotent:**
  \[ S \cup S = S \]
  \[ S \cap S = S \]

• **Complementation:**
  \[ (S')' = S \]
Set Identities (Cont.)

- **Commutative Law:**
  \[ S \cup T = T \cup S \]
  \[ S \cap T = T \cap S \]

- **Associative Law:**
  \[ S \cup (T \cup V) = (S \cup T) \cup V \]
  \[ S \cap (T \cap V) = (S \cap T) \cap V \]
Set Identities (Cont.)

- **Distributive Law:**

  \[ S \cup (T \cap V) = (S \cup T) \cap (S \cup V) \]

  \[ S \cap (T \cup V) = (S \cap T) \cup (S \cap V) \]

- **Absorption:**

  \[ S \cup (S \cap T) = S \]

  \[ S \cap (S \cup T) = S \]
DeMorgan's Laws

$$(S \cup T)' = S' \cap T'$$

Boolean logic version:
$$(X \land Y)' = X' \lor Y'$$
$$(X \lor Y)' = X' \land Y'$$
Generalized $\bigcup$ and $\bigcap$

- $\bigcup_{1 \leq i \leq n} S_i = S_1 \cup S_2 \cup S_3 \cup \ldots \cup S_n$
  
  $$= \{x \mid \exists i \; 1 \leq i \leq n \; \exists x \in S_i\}$$

- $\bigcap_{1 \leq i \leq n} S_i = S_1 \cap S_2 \cap S_3 \cap \ldots \cap S_n$
  
  $$= \{x \mid \forall i \; 1 \leq i \leq n \Rightarrow x \in S_i\}$$
Set Representation

• \( U = \{x_1, x_2, x_3, x_4, \ldots, x_{n-1}, x_n\} \)

Ex: \( S = \{x_1, x_3, x_n\} \)

bits: \(1\ 0\ 1\ 0\ \ldots\ \ 0\ 0\ 1\)

1010000...01 encodes \( \{x_1, x_3, x_n\} \)
0111000...00 encodes \( \{x_2, x_3, x_4\} \)

• “or” yields union:

\[
\begin{align*}
1010000...01 \land 0111000...00 & = 1111000...01 \\
\{x_1, x_3, x_n\} \land \{x_2, x_3, x_4\} & = \{x_1, x_2, x_3, x_4, x_n\}
\end{align*}
\]

• “and” yields intersection:

\[
\begin{align*}
1010000...01 \lor 0111000...00 & = 1111000...01 \\
\{x_1, x_3, x_n\} \lor \{x_2, x_3, x_4\} & = \{x_1, x_2, x_3, x_4, x_n\}
\end{align*}
\]
• Set closure: WRT operation $\Delta$

$$\forall x, y \in S \Rightarrow x \Delta y \in S$$

\[
\begin{array}{ccc}
& x & \Delta y \\
\cdot & y & x \\
\end{array}
\]

• Ex: $\mathbb{R}$ is closed under addition since $x, y \in \mathbb{R} \Rightarrow x + y \in \mathbb{R}$

Abbreviations

• WRT “with respect to”
• WLOG “without loss of generality”

"When ideas fail, words come in very handy."
- Goethe (1749-1832)
Cartesian Product

• **Ordered n-tuple**: element sequence
  
  Ex: \((2,3,5,7)\) is a 4-tuple

• **Tuple equality**:
  
  \((a,b) = (x,y) \iff (a=x) \land (b=y)\)
  
  Generally: \((a_i) = (x_i) \iff \forall i \ a_i = x_i\)

• **Cross-product**: ordered tuples
  
  \[ S \times T = \{(s,t) \mid s \in S, \ t \in T\} \]

  Ex: \(\{1, 2, 3\} \times \{a, b\} = \{(1,a),(1,b),(2,a),(2,b),(3,a),(3,b)\}\)

  Generally, \(S \times T \neq T \times S\)
• Generalized cross-product:

\[ S_1 \times S_2 \times \ldots \times S_n = \{(x_1, \ldots, x_n) | x_i \in S_i, 1 \leq i \leq n\} \]

\[ T^i = T \times T^{i-1} \]

\[ T^1 = T \]

• Euclidean plane = \( \mathbb{R} \times \mathbb{R} = \mathbb{R}^2 \)

• Euclidean space = \( \mathbb{R} \times \mathbb{R} \times \mathbb{R} = \mathbb{R}^3 \)

• Russel’s paradox: set of all sets that do not contain themselves:

\[ \{S | S \not\in S \} \]

Q: Does S contain itself??
Functions

- **Function**: mapping $f:S \rightarrow T$

  Domain $S$

  Range $T$

- **k-ary**: has $k$ “arguments”
- **Predicate**: with range $= \{ \text{true, false} \}$
Function Types

• **One-to-one function:** "1-1"
  \[ a, b \in S \land a \neq b \Rightarrow f(a) \neq f(b) \]
  Ex: \( f: \mathbb{R} \to \mathbb{R}, f(x) = 2x \) is 1-1
  \( g(x) = x^2 \) is not 1-1

• **Onto function:**
  \[ \forall t \in T \ \exists s \in S \ \exists f(s) = t \]
  Ex: \( f: \mathbb{Z} \to \mathbb{Z}, f(x) = 13 - x \) is onto
  \( g(x) = x^2 \) is not onto
1-to-1 Correspondence

- **1-to-1 correspondence**: \( f: S \leftrightarrow T \)

  \( f \) is both 1-1 and onto

Ex: \( f: \mathbb{R} \leftrightarrow \mathbb{R} \ \& \ f(x) = x \) (identity)

\( h: \mathbb{N} \leftrightarrow \mathbb{Z} \ \& \ h(x) = \frac{x-1}{2} \), \( x \) odd,

\( -\frac{x}{2} \), \( x \) even.
• **Inverse function:**

\[ f: S \to T \quad f^{-1}: T \to S \]

\[ f^{-1}(t) = s \quad \text{if} \quad f(s) = t \]

**Ex:** \( f(x) = 2x \quad f^{-1}(x) = x/2 \)

• **Function composition:**

\( \beta: S \to T, \alpha: T \to V \)

\[ \Rightarrow \quad (\alpha \cdot \beta)(x) = \alpha(\beta(x)) \]

\( (\alpha \cdot \beta): S \to V \)

**Ex:** \( \beta(x) = x+1 \quad \alpha(x) = x^2 \)

\( (\alpha \cdot \beta)(x) = x^2 + 2x + 1 \)
Thm: \((f \cdot f^{-1})(x) = (f^{-1} \cdot f)(x) = x\)
Set Cardinality

• **Cardinality:** \(|S| = \#\text{elements in } S\)

  Ex: \(|\{a, b, c\}| = 3\)
  \(|\{p \mid p \text{ prime} < 9\}| = 4\)
  \(|\emptyset| = 0\)
  \(|\{\{1, 2, 3, 4, 5\}\}| = ?\)

• **Powerset:** \(2^S = \text{set of all subsets}\)

  \(2^S = \{T \mid T \subseteq S\}\)

  Ex: \(2^{\{a, b\}} = \{\{\}, \{a\}, \{b\}, \{a, b\}\}\)

  Q: What is \(2^\emptyset\)?
Theorem: \(|2^S| = 2^{|S|}\)

Proof:

“Sometimes when reading Goethe, I have the paralyzing suspicion that he is trying to be funny.”

- Guy Davenport
Generalized Cardinality

- **S is at least as large as T:**
  \[ |S| \geq |T| \implies \exists f : S \rightarrow T, f \text{ onto} \]
  i.e., “S covers T”

  Ex: \( r : \mathbb{R} \rightarrow \mathbb{Z}, r(x) = \text{round}(x) \)

  \[ \implies |\mathbb{R}| \geq |\mathbb{Z}| \]

- **S and T have same cardinality:**
  \[ |S| = |T| \implies |S| \geq |T| \land |T| \geq |S| \]
  or
  \[ \exists \text{ 1-1 correspondence } S \leftrightarrow T \]

- **Generalizes finite cardinality:**
  \[ \{1, 2, 3, 4, 5\} \geq \{a, b, c\} \]
Infinite Sets

- **Infinite set**: \(|S| > k \ \forall k \in \mathbb{Z}\)
  
or
  \(\exists 1-1 \text{ corres. } f:S \leftrightarrow T, \ S \subset T\)

Ex: \(\{p \mid p \text{ prime}\}\), \(\mathbb{R}\)

- **Countable set**: \(|S| \leq |\mathbb{N}|\)

Ex: \(\emptyset\), \(\{p \mid p \text{ prime}\}\), \(\mathbb{N}, \mathbb{Z}\)

- **S is strictly smaller than T**: \(|S| < |T| \ \Rightarrow \ |S| \leq |T| \wedge |S| \neq |T|\)

- **Uncountable set**: \(|\mathbb{N}| < |S|\)

Ex: \(|\mathbb{N}| < \mathbb{R}\)

\(|\mathbb{N}| < [0,1] = \{x \mid x \in \mathbb{R}, 0 \leq x \leq 1\}\)
Thm: \( \exists \) 1-1 correspondence \( \mathbb{Q} \leftrightarrow \mathbb{N} \)

Pf (dove-tailing):

\[
\begin{array}{cccccccc}
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
\hline
1 & 2 & 3 & 4 & 5 & 6 & 6 & 6 \\
6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & 2 & 3 & 4 & 5 & 6 \\
\end{array}
\]
Thm: \(|\mathbb{R}| > |\mathbb{N}|\)

Pf (diagonalization):

Assume \(\exists 1-1\) corres. \(f: \mathbb{R} \leftrightarrow \mathbb{N}\)

Construct \(x \in \mathbb{R}\):

\[
\begin{align*}
  f(1) &= 2.718281828 \ldots \rightarrow 8 \\
  f(2) &= 1.414213562 \ldots \rightarrow 2 \\
  f(3) &= 1.618033989 \ldots \rightarrow 9
\end{align*}
\]

\(x = 0.829 \ldots \neq f(K) \ \forall K \in \mathbb{N}\)

\(\Rightarrow f\) not a 1-1 correspondence

\(\Rightarrow\) contradiction

\(\Rightarrow \mathbb{R}\) is uncountable
Q: Is $|2^Z| = |\mathbb{R}|$?
Q: Is $|\mathbb{R}| > |[0,1]|$?
Thm: any set is "smaller" than its powerset.

\[ |S| < |2^S| \]
Infinities

- $|\mathbb{N}| = \aleph_0$
- $|\mathbb{R}| = \aleph_1$
- $\aleph_0 < \aleph_1 = 2^{\aleph_0}$

- “Continuum Hypothesis”

$\exists ? \omega \in \aleph_0 < \omega < \aleph_1$

Independent of the axioms!

[Cohen, 1966]

- Axiom of choice [Godel 1938]
- Parallel postulate
Infinity Hierarchy

- $\aleph_i < \aleph_{i+1} = 2^{\aleph_i}$

- $0, 1, 2, \ldots, k, k+1, \ldots, \aleph_0,$

- $\aleph_1, \aleph_2, \ldots, \aleph_k, \aleph_{k+1}, \ldots,$

- $\aleph_0, \aleph_1, \ldots, \aleph_k, \aleph_{k+1}, \ldots$

• First inaccessible infinity: $\omega$...

For an informal account on infinities, see e.g.: Rucker, *Infinity and the Mind*, Harvester Press, 1982.
**Thm**: # algorithms is countable.
**Pf**: sort programs by size:

```
"main(){}"
.
"main(){int k; k=7;}" 
.
"<all of UNIX>"
.
"<Windows XP>"
.
"<intelligent program>"
.
```

⇒ # algorithms is countable!
**Thm:** # of functions is uncountable.

**Pf:** Consider 0/1-valued functions (i.e., functions from \( N \) to \( \{0,1\} \)):
\[
\{(1,0), (2,1), (3,1), (4,0), (5,1), \ldots \}
\]
\[
\Rightarrow \{ 2, 3, 5, \ldots \} \in 2^N
\]

So, every subset of \( N \) corresponds to a different 0/1-valued function

\[|2^N| \text{ is uncountable (why?)}\]

\[\Rightarrow \# \text{ functions is uncountable!}\]
Thm: most functions are uncomputable!

Pf: \# algorithms is countable
    \# functions is not countable

⇒ ∃ more functions than
    algorithms / programs!

⇒ some functions do not have
    algorithms!

Ex: The halting problem

Given a program P and input I, does P halt on I?

Def: H(P,I) = 1 if P halts on I
    0 otherwise
The Halting Problem

H: Given a program P and input I, does P halt on I? i.e., does P(I)↓?

Thm: H is uncomputable

Pf: Assume subroutine S solves H.

Construct:
Analyze:

\[ S'(S') \downarrow \Rightarrow S'(S') \uparrow \]
\[ S'(S') \uparrow \Rightarrow S'(S') \downarrow \]

so, \[ S'(S') \uparrow \iff S'(S') \downarrow \]
a contradiction!

\[ \Rightarrow S \text{ does not correctly compute } H \]

But \( S \) was an arbitrary subroutine, so
\[ \Rightarrow H \text{ is not computable!} \]
Discrete Probability

Sample space: set of possible outcomes

Event E: subset of sample space S

Probability p of an event: \(|E| / |S|\)

- \(0 \leq p \leq 1\)
- \(p(\text{not}(E)) = 1 - p(E)\)
- \(p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)\)

Ex: two dice yielding total of 9

\(E=\{(3,6),(4,5),(5,4),(6,3)\}\)

\(S=\{1,2,3,4,5,6\} \times \{1,2,3,4,5,6\}\)

\(p(E) = |E|/|S| = 4/36 = 1/9\)
General Probability

Outcome $x_i$ is assigned probability $p(x_i)$

- $0 \leq p(x_i) \leq 1$
- $\sum p(x_i) = 1$
- $E = \{a_1, a_2, \ldots, a_m\} \rightarrow p(E) = \sum p(a_i)$
- $p(\text{not}(E)) = 1 - p(E)$
- $p(E_1 \cup E_2) = p(E_1) + p(E_2) - p(E_1 \cap E_2)$

Conditional Probability

$p(E \mid F) = \text{probability of } E \text{ given } F$

$p(E \cap F) = p(F) \cdot p(E \mid F)$
Ex: what is the probability of two siblings being both male, given that one of them is male?

Let \((x,y)\) be the two siblings
Sample space: \(\{(m,m),(m,f),(f,m),(f,f)\}\)

Let \(E\) = both are male
\[= \{(m,m)\}\]

Let \(F\) = at least one is male
\[= \{(m,m),(m,f),(f,m)\}\]

\(E \cap F\) = \(\{(m,m)\}\)
\[= \text{both are male}\]

\[p(E \cap F) = p(F) \cdot p(E \mid F)\]

\[p(E \mid F) = \frac{p(E \cap F)}{p(F)}\]
\[= \frac{1/4}{3/4} = \frac{1}{3}\]
Relations

Relation: a set of “ordered tuples”

Ex: \{ (a,1), (b,2), (b,3) \}

“<” \{ (x,y) \mid x, y \in \mathbb{Z}, x < y \}

**Reflexive:** \( x \bowtie x \ \forall x \)

**Symmetric:** \( x \bowtie y \Rightarrow y \bowtie x \)

**Transitive:** \( x \bowtie y \land y \bowtie z \Rightarrow x \bowtie z \)

**Antisymmetric:** \( x \bowtie y \Rightarrow \neg(y \bowtie x) \)

Ex: \( \leq \) is reflexive
    transitive
    not symmetric
Equivalence Relations

Def: reflexive, symmetric, & transitive

Ex: standard equality “=”

\[ x = x \]
\[ x = y \implies y = x \]
\[ x = y \land y = z \implies x = z \]

Partition - disjoint equivalence classes:
Closures

• Transitive closure of $\heartsuit$: TC
  smallest superset of $\heartsuit$ satisfying
  \[ x \heartsuit y \land y \heartsuit z \Rightarrow x \heartsuit z \]

  Ex: “predecessor”
  \[ \{(x-1,x) \mid x \in \mathbb{Z}\} \]
  TC(predecessor) is “<” relation

• Symmetric closure of $\heartsuit$:
  smallest superset of $\heartsuit$ satisfying
  \[ x \heartsuit y \Rightarrow y \heartsuit x \]
Algorithms

• Existence
• Efficiency

Analysis

• Correctness
• Time
• Space
• Other resources

Worst case analysis
(as function of input size |w|)

Asymptotic growth: $O \ \Omega \ \Theta \ o$
Upper Bounds

\[ f(n) = O(g(n)) \iff \exists c,k > 0 \quad \exists |f(n)| \leq c \cdot |g(n)| \quad \forall n > k \]

\[ \lim_{n \to \infty} f(n) / g(n) \text{ exists} \]

“\( f(n) \) is big-O of \( g(n) \)”

Ex: \( n = O(n^2) \)

\[ 33n + 17 = O(n) \]
\[ n^8 - n^7 = O(n^{123}) \]
\[ n^{100} = O(2^n) \]
\[ 213 = O(1) \]
Lower Bounds

\[ f(n) = \Omega(g(n)) \iff g(n) = O(f(n)) \]

\[
\lim_{n \to \infty} \frac{g(n)}{f(n)} \text{ exists}
\]

“\( f(n) \) is Omega of \( g(n) \)”

Ex: \( 100n = \Omega(n) \)

\( 33n + 17 = \Omega(\log n) \)

\( n^8 - n^7 = \Omega(n^8) \)

\( 213 = \Omega(1/n) \)

\( 1 = \Omega(213) \)
Tight Bounds

\[ f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \land g(n) = O(f(n)) \]

“\( f(n) \) is Theta of \( g(n) \)”

Ex: 100n = \( \Theta(n) \)

\[ 33n + 17 + \log n = \Theta(n) \]

\[ n^8 - n^7 - n^{-13} = \Theta(n^8) \]

\[ 213 = \Theta(1) \]

\[ 3 + \cos(2^n) = \Theta(1) \]
Loose Bounds

\[ f(n) = o(g(n)) \iff f(n) = O(g(n)) \land f(n) \neq \Omega(g(n)) \]

\[ \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \]

“\( f(n) \) is little-o of \( g(n) \)”

Ex: \( 100n = o(n \log n) \)

\[ 33n + 17 + \log n = o(n^2) \]
\[ n^8 - n^7 - n^{-13} = o(2^n) \]
\[ 213 = o(\log n) \]
\[ 3 + \cos(2^n) = o(\sqrt{n}) \]
Growth Laws

Let \( f_1(n) = O(g_1(n)) \) and \( f_2(n) = O(g_2(n)) \)

\[ \text{Thm: } f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n))) \]

\[ \text{Thm: } f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n)) \]

\[ \text{Thm: } n^k = O(c^n) \quad \forall \ c, k > 0 \]

Ex: \( n^{1000} = O(1.001^n) \)
Recurrences

\[ T(n) = a \cdot T(n/b) + f(n) \]

let \( c = \log_b a \)

Thm:

\[ f(n) = O(n^{c-\varepsilon}) \Rightarrow T(n) = \Theta(n^c) \]
\[ f(n) = \Theta(n^c) \Rightarrow T(n) = \Theta(n^c \log n) \]
\[ f(n) = \Omega(n^{c+\varepsilon}) \land a \cdot f(n/b) \leq d \cdot f(n) \]

\[ \forall d < 1, n > n_0 \Rightarrow T(n) = \Theta(f(n)) \]

Ex: \( T(n) = 9T(n/3) + n \Rightarrow T(n) = \Theta(n^2) \)

\[ T(n) = T(2n/3) + 1 \Rightarrow T(n) = \Theta(\log n) \]
Pigeon-Hole Principle

If $N+1$ objects are placed into $N$ boxes $\Rightarrow \exists$ a box with 2 objects.

If $M$ objects are placed into $N$ boxes & $M>N \Rightarrow \exists$ box with $\left\lfloor \frac{M}{N} \right\rfloor$ objects.

• Useful in proofs & analyses
Stirling's Formula

\[ n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-2) \cdot (n-1) \cdot n \]

\[ n! = \sqrt{2\pi n} \cdot \left( \frac{n}{e} \right)^n \cdot (1 + \Theta\left(\frac{1}{n}\right)) \]

\[ n! \approx \left( \frac{n}{e} \right)^n \]

\[ \log(n!) = O(n \log n) \]

- Useful in analyses and bounds
Data Structures

• What is a "data structure"?

• Operations:
  • Initialize
  • Insert
  • Delete
  • Search
  • Min/max
  • Successor/Predecessor
  • Merge
Arrays

• Sequence of "indexible" locations

```
1 2 3 4 5 6 7 ...
```

• Unordered:
  • O(1) to add
  • O(n) to search
  • O(n) for min/max

• Ordered:
  • O(n) to add
  • O(log n) to (binary) search
  • O(1) for min/max
Stacks

- LIFO (last-in first-out)

- Operations: push/pop (O(1) each)

- Can not access "middle"

- Analogy: trays at Cafeteria

- Applications:
  - Compiling / parsing
  - Dynamic binding
  - Recursion
  - Web surfing
Queues

• FIFO (first-in first-out)

  in → [] → out

• Operations: push/pop (O(1) each)

• Can not access "middle"

• Analogy: line at your Bank

• Applications:
  • Scheduling
  • Operating systems
  • Simulations
  • Networks
Linked Lists

- Successor pointers

- Types:
  - Singly linked
  - Doubly linked
  - Circular

- Operations:
  - Add: $O(1)$ time
  - Search: $O(n)$ time
  - Delete: $O(1)$ time (if known)
Trees

- Parent/children pointers

- Binary/N-ary

- Ordered/unordered

- Height-balanced:
  - AVL
  - B-trees
  - Red-black
  - $O(\log n)$ worst-case time
Tree Traversals

- **pre-order:**
  1) process node
  2) visit children

\[
\Rightarrow \quad \text{c b a e d f}
\]

- **post-order:**
  1) visit children
  2) process node

\[
\Rightarrow \quad \text{a b d f e c}
\]

- **in-order:**
  1) visit left-child
  2) process node
  3) visit right-child

\[
\Rightarrow \quad \text{a b c d e f}
\]
Heaps

• A tree where all of a node’s children have smaller “keys”

• Can be implemented as a binary tree

• Can be implemented as an array

• Operations:
  • Find max: $O(1)$ time
  • Add: $O(\log n)$ time
  • Delete: $O(\log n)$ time
  • Search: $O(n)$ time
Hash Tables

- Direct access
- Hash function
- Collision resolution:
  - Chaining
  - Linear probing
  - Double hashing
- Universal hashing
- $O(1)$ average access
- $O(n)$ worst-case access

Q: How can worst-case access time be improved to $O(\log n)$?
Fact: almost half of all CPU cycles are spent on sorting!!

- Input: array X[1..n] of integers
  Output: sorted array
- Decision tree model

**Thm:** Sorting takes $\Omega(n \log n)$ time

**Pf:** $n!$ different permutations

$\Rightarrow$ decision tree has $n!$ leaves

$\Rightarrow$ tree height is: $\log(n!)$

$\quad > \log((n/e)^n)$

$\quad = \Omega(n \log n)$
Sort Properties

- Worst case?
- Average case?
- In practice?
- Input distribution?
- Randomized?
- Stability?
- In-Situ?
- Stack depth?
- Internal vs. external?
• **Bubble Sort:**

For $k=1$ to $n$
   For $i=1$ to $n-1$
      If $X[i+1] > X[i]$
         Then Swap($X,i,i+1$)

$\Rightarrow \Theta(n^2)$ time

• **Insertion Sort:**

For $i=1$ to $n-1$
   For $j=i+1$ to $n$
      If $X[j] > X[i]$ Then Swap($X,i,j$)

$\Rightarrow \Theta(n^2)$ time
• Quicksort:

Quicksort(X,i,j)
    If i<j Then  p=Partition(X,i,j)
        QuickSort(X,i,p)
        QuickSort(X,p+1,j)

⇒O(n log n) time (ave-case)

• C.A.R. Hoare, 1962
• Good news: usually best in practice
• Bad news: worst-case O(n^2) time
• Usually avoids worst-case
• Only beats O(n^2) sorts for n>40
• **Merge Sort:**

\[
\text{MergeSort}(X, i, j) \\
\begin{align*}
\text{if } i &< j \text{ then } m = \lfloor (i+j)/2 \rfloor \\
\text{MergeSort}(X, i, m) \\
\text{MergeSort}(X, m+1, j) \\
\text{Merge}(X, i, m, j)
\end{align*}
\]

\[
T(n) = 2T(n/2) + n
\]

\[\Rightarrow \Theta(n \log n) \text{ time}\]

---

• **Heap Sort:**

\[
\text{InitHeap} \\
\text{For } i = 1 \text{ to } n \text{ HeapInsert}(X(i)) \\
\text{For } i = 1 \text{ to } n \text{ M=HeapMax} \\
\text{Print}(M) \\
\text{HeapDelete}(M)
\]

\[\Rightarrow \Theta(n \log n) \text{ time}\]
• **Counting Sort:**

Assumes integers in small range 1..k

For i=1 to k C[i]=0
For i=1 to k C[X[i]]++
For i=1 to k
   If C[i]>0 Then print(i) C[i] times

\[\Theta(n) \text{ time (worst-case)}\]

• **Radix Sort:**

Assumes d digits in range 1..k

For i=1 to d StableSort(X on digit i)

\[O(dn+kd) \text{ time (worst-case)}\]
• **Bucket Sort:**

Assumes uniform inputs in range 0..1

For i=1 to n
   Insert X[i] into Bucket ⌊n \cdot X[i]⌋

For i=1 to n  Sort Bucket i

Concat contents of Buckets 1 thru n

⇒ O(n) time (expected)

O(Sort) time (worst)
Order Statistics

• **Exact** comparison count

• Minimum element

\[ k = X[1] \]

For \( i = 2 \) to \( n \)

\[ \text{If } X[i] < k \text{ Then } k = X[i] \]

\[ \Rightarrow n - 1 \text{ comparisons} \]

**Thm:** Min requires \( n - 1 \) comparisons.

**Proof:**
• **Min and Max:**

(a) Compare all pairs  
(b) Find Min of min’s of all pairs  
(c) Find Max of max’s of all pairs

⇒ \[ \frac{n}{2} + \frac{n}{2} + \frac{n}{2} = \frac{3n}{2} \] comparisons

**Thm:** Min & Max require \(\frac{3n}{2}\) comparisons.  
**Pf:** Represent known info by four sets:

<table>
<thead>
<tr>
<th>Unknown</th>
<th>Not Min</th>
<th>Not Max</th>
<th>Neither</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

Initial:  
- n
- 0
- 0
- 0

Final:  
- 0
- 1
- 1
- n-2

Track movement of elements between sets.
Effect of comparisons:

<table>
<thead>
<tr>
<th>Origin</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A&amp;A C&amp;B</td>
</tr>
<tr>
<td>A&amp;B</td>
<td>C&amp;B B&amp;D</td>
</tr>
<tr>
<td>A&amp;C</td>
<td>C&amp;D B&amp;C</td>
</tr>
<tr>
<td>A&amp;D</td>
<td>C&amp;D B&amp;D</td>
</tr>
<tr>
<td>B&amp;B</td>
<td>D&amp;B B&amp;D (2)</td>
</tr>
<tr>
<td>B&amp;C</td>
<td>D&amp;D B&amp;C</td>
</tr>
<tr>
<td>B&amp;D</td>
<td>D&amp;D B&amp;D</td>
</tr>
<tr>
<td>C&amp;C</td>
<td>C&amp;D D&amp;C (3)</td>
</tr>
<tr>
<td>C&amp;D</td>
<td>C&amp;D D&amp;D</td>
</tr>
<tr>
<td>D&amp;D</td>
<td>D&amp;D D&amp;D</td>
</tr>
</tbody>
</table>

- Going from A to D forces passing through B or C
- "Emptying" A into B&C takes n/2 comparisons (1)
- "Almost emptying" B takes n/2-1 comparisons (2)
- "Almost emptying" C takes n/2-1 comparisons (3)
- Other moves will not reach the "final state" faster
- Total comparisons required: 3n/2-2
Problem: Find Max and next-to-Max using least # of comparisons.
Selection

• Not harder than median-finding (why?)
• Randomized $i^{th}$-Selection
  (return the $i^{th}$-largest element in $X[p..r]$)

Select($X,p,r,i$)
  If $p=r$ Then Return($X[p]$)
  $q=\text{RandomPartition}(X,p,r)$
  $k=q-p+1$
  If $i \leq k$ Then Return($\text{Select}(X,p,q,i)$)
  Else Return($\text{Select}(X,q+1,r,i-k)$)

$\Rightarrow$ O($n$) time (ave-case)
Deterministic $i^{th}$-Selection

[Blum, Floyd, Pratt, Rivest, Tarjan; 1973]

- Partition input into $n/5$ groups of 5 each
- Compute median of each group
- Compute median of medians (recursively)

- Compute median of medians (recursively)
- Eliminate $3n/10$ elements & recurse on rest

\[ T(n) = T(n/5) + T(7n/10) + O(n) \]
\[ = T(2n/10) + T(7n/10) + O(n) \]
\[ \leq T(9n/10) + O(n) \text{ since } T(n) = \Omega(n) \]

\[ \Rightarrow T(n) = O(n) \]
Problem: Find in O(n) time the majority element (i.e., occurring $\geq n/2$ times, if any).

a) Using "<",">","="

b) Using "=" only (i.e., no "order")
Graphs

- A special kind of relation

Graphs can model:
  - Common relationships
  - Communication networks
  - Dependency constraints
  - Reachability information
  + many more practical applications!

**Graph** $G=(V,E)$: set of vertices $V$, and a set of edges $E \subseteq V \times V$

Pictorially: nodes & lines
Undirected Graphs

Def: edges have no direction

- Example of undirected graph:

V=\{a, b, c, d, e\}
E=\{(c, a), (c, b), (c, d), (c, e), (a, b), (b, d), (d, e)\}
Directed Graphs

Def: edges **have** direction

- Example of directed graph:

V={a,b,c,d,e}
E={(a,b),(a,c),(b,c),(b,d), (d,c),(d,e),(c,e)}
Graph Terminology

Graph $G=(V,E)$, $E \subseteq V \times V$

- node $\equiv$ vertex
- edge $\equiv$ arc

Vertices $u,v \in V$ are neighbors in $G$ iff $(u,v)$ or $(v,u)$ is an edge of $G$

Ex: $a$ & $b$ are neighbors
    $a$ & $e$ are not neighbors
Undirected Node Degree

Degree in undirected graphs:

Degree(v) = # of adjacent (incident) edges to vertex v in G

Ex: \( \text{deg}(c)=4 \quad \text{deg}(f)=0 \)
Directed Node Degree

Degree in directed graphs:

**In-degree** \(v) = \# \text{ of incoming edges} \\
**Out-degree** \(v) = \# \text{ of outgoing edges}

Ex: \( \text{in-deg}(c) = 3 \) \quad \text{out-deg}(c) = 1 \\
\( \text{in-deg}(f) = 0 \) \quad \text{out-deg}(f) = 0
Q: Show that at any party there is an even number of people who shook hands an odd number of times.
Complete graph $K_n$ contains all edges i.e., $E = \{ \{u,v\} \in V \times V \mid u \neq v \}$

Q: How many edges are there in $K_n$?

Subgraph of $G$ is $G'=(V',E')$ where $V' \subseteq V$ and $E' \subseteq E$

Q: Give a (non-trivial) lower bound on the number of graphs over $n$ vertices.
Paths in Graphs

Undirected path in a graph:

A graph is connected iff there is a path between any pair of nodes:
Directed path in a graph:

Graph is **strongly connected** iff there is a directed path between **any** node pair:

Ex: connected but not **strongly**:
A cycle in a graph:

A tree is an acyclic graph.

Tree T=(V’,E’) spans G=(V,E) if T is a connected subgraph with V’=V
Q: How many edges are there in a tree over n vertices?

Q: Is the # of distinct spanning trees in a graph G always polynomial in |G|?
Graph Traversals

Breadth-first search:

Depth-first search:

$O(E+V)$ time for either BFS or DFS

Yields a spanning tree for the graph
Topological Sort

Given a digraph, list vertices so that all edges point/direct to the right:

Can be done in $O(E+V)$ time

Application: scheduling w/constraints
Weighted Graphs

Each edge has a weight: \( w: E \rightarrow \mathbb{Z} \)

Weights can model many things:

- Distances / lengths
- Speed / time
- Costs

\[
\text{Cost}(G) = \text{sum of edge costs}
\]

Find a shortest / least-expensive subgraph with a given property
Graph Representation

Adjacency list:

1: (a) → b → c
2: (b) → a → d
3: (c) → a
4: (d) → b

Adjacency matrix:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Minimum Spanning Trees
Prim’s MST Algorithm

\[ T = v_0 \]

Until T spans all nodes do
   Select nodes \( x \in T \), \( y \notin T \)
   \( w/\text{min cost}(x,y) \)
   Add edge \((x,y)\) to T

Return T

- Time complexity: \( O(E \log E) \)
- Kruskal: \( O(E \log V) \)
- Fibonacci heaps: \( O(E + V \log V) \)
Shortest Paths Trees

Diagram of a graph with labeled edges and a shortest path tree highlighted.
Dijkstra’s Single-Source Shortest paths Algorithm

\[
T = v_0 \\
\text{Until } T \text{ spans all nodes do} \\
\quad \text{Select nodes } x \in T, y \notin T \\
\quad \quad \text{w/min cost}(x,y) + \text{dist}(v_0,x) \\
\quad \text{Add edge } (x,y) \text{ to } T \\
\text{Return } T
\]

- Time complexity: $O(V^2)$
- All pairs: $O(V^3)$
Cost-Radius Tradeoffs


\[
\text{Signal delay} \uparrow \Rightarrow \text{Performance} \downarrow
\]

- Source $\rightarrow$ sink pathlength $\propto$ delay
  
  $\Rightarrow$ Avoid long paths

- Capacitive delay / building cost
  
  $\Rightarrow$ Minimize total wirelength
Possible Trees

MST:

SPT:

?
Definitions

Input: pointset with distinguished source

\textit{ptset radius} \(R\): max source-sink dist

\textit{tree radius}: max source-sink pathlength
Problem Formulation

Given a pointset $P$, $\varepsilon \geq 0$, find min-cost tree $T$ with $r(T) \leq (1+\varepsilon) \cdot R$

**Tradeoff:** $\varepsilon$ trades off radius and tree cost

$\varepsilon = 0 \Rightarrow \text{"Shortest Path Tree"}$

$\varepsilon = \infty \Rightarrow \text{Minimum Spanning Tree}$
Arbitrary $\varepsilon \Rightarrow$ hybrid construction

- Unifies Prim and Dijkstra!
Bounded Radius MSTs

Goal: \( \text{cost} \approx \text{cost(MST)} \)

\( \text{radius} \approx r(\text{SPT}) \)

- Let \( Q = \text{MST} \)

- Let \( L \) be tour of \( \text{MST} \):

```
s
  a
  |
  b  c
  |
  d
  |
  e
  |
  f
  |
  g
  |
  h
```
• Traverse L

• A = running total of edge costs

• If $A > \varepsilon \cdot R$ Then $A = 0$

\[
Q = Q \cup \text{minpath}_G(s, L_i)
\]

• Final **routing** tree is $\text{SPT}_Q$

\[
L = \text{MST tour}
\]

Shortest paths added
\[ \text{dist}_T(s,v) \leq \text{dist}_G(s,v_i) + \text{dist}_L(v_i,v) \]
\[ \leq R + \varepsilon \cdot R = (1 + \varepsilon) \cdot R \]
\[ \Rightarrow r(T) \leq (1 + \varepsilon) \cdot R \]
\[ \text{cost}(T) \leq \text{cost}(\text{MST}_G) + \frac{\text{cost}(L)}{\varepsilon \cdot R} \cdot R \]

\[ = \text{cost}(\text{MST}_G) + \frac{2 \cdot \text{cost}(\text{MST}_G)}{\varepsilon} \]

\[ = (1 + \frac{2}{\varepsilon}) \cdot \text{cost}(\text{MST}_G) \]

\[ \Rightarrow \text{cost}(T) \leq (1 + \frac{2}{\varepsilon}) \cdot \text{cost}(\text{MST}_G) \]
Bounded Radius MST Algorithm

Compute $\text{MST}_G$ and $\text{SPT}_G$

$E' = \text{edges of } \text{MST}_G$

$Q = (V,E')$

$L = \text{depth-first tour of } \text{MST}_G$

$A = 0$

For $i = 2$ to $|L|$

$A = A + \text{cost}(L_{i-1}, L_i)$

If $A > \varepsilon \cdot R$ Then

$E' = E' \cup \text{minpath}_G(s, L_i)$

$A=0$

$T = \text{SPT}_Q$

Input: $G=(V,E)$, source $s$, radius $R$, $0 \leq \varepsilon$

Output: $T = \text{routing tree with}$

$\text{cost}(T) \leq (1+\frac{2}{\varepsilon}) \cdot \text{cost(MST}_G)$

$r(T) \leq (1+\varepsilon) \cdot R$
Steiner Trees
Bounded Radius Steiner Trees

Given weighted graph $G=(V,E)$, node subset $N$, source $s \in N$, and $0 \leq \varepsilon$, find min-cost tree $T$ spanning $N$, with $r(T) \leq (1+\varepsilon) \cdot r(N)$

- NP-complete
Bounded Radius Steiner Trees

- Can use *any* low-cost spanning tree

- Use [KMB, 1981] to span $N$ (cost $\leq 2 \cdot \text{opt}$)

- Run previous algorithm

$$\Rightarrow \text{cost}(T) \leq 2 \cdot (1 + \frac{2}{\epsilon}) \cdot \text{opt}$$
Geometry Helps

- Add Steiner points when $A = 2\varepsilon \cdot R$

- Use bounds on MST/Steiner ratio

<table>
<thead>
<tr>
<th>Tree type</th>
<th>Graph type</th>
<th>Radius bound</th>
<th>Cost bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>spanning</td>
<td>arbitrary</td>
<td>$(1+\varepsilon) \cdot R$</td>
<td>$(1 + 2/\varepsilon) \cdot \text{MST}$</td>
</tr>
<tr>
<td>Steiner</td>
<td>arbitrary</td>
<td>$(1+\varepsilon) \cdot R$</td>
<td>$2 \cdot (1+ 2/\varepsilon) \cdot \text{opt}$</td>
</tr>
<tr>
<td>Steiner</td>
<td>Manhattan</td>
<td>$(1+\varepsilon) \cdot R$</td>
<td>$\frac{3}{2} (1+1/\varepsilon) \cdot \text{opt}$</td>
</tr>
<tr>
<td>Steiner</td>
<td>Euclidean</td>
<td>$(1+\varepsilon) \cdot R$</td>
<td>$\frac{2}{\sqrt{3}} \cdot (1+1/\varepsilon) \cdot \text{opt}$</td>
</tr>
</tbody>
</table>
Experimental Results

![Graph showing experimental results for net size and cost ratio.](image)

- **r(T) / r(MST)**
  - $\varepsilon = 0.25$
  - $\varepsilon = 1.00$
  - $\varepsilon = 2.00$

- **cost(T) / cost(MST)**
  - $\varepsilon = 0.25$
  - $\varepsilon = 1.00$
  - $\varepsilon = 2.00$

**Net size**: 5, 8, 10, 15, 25

**MST** is represented as a line connecting all points, indicating the minimum spanning tree. **SPT** shows a different trend compared to MST. The graphs illustrate the performance of different parameters on the ratio and cost ratio with respect to the net size.
NP-Completeness

- Tractability
- Polynomial time
- Computation vs. verification
- Non-determinism
- Encodings
- Transformation & reducibilities
- P vs. NP
- "completeness"
A problem L is NP-hard if:
1) all problems in NP reduce to L in polynomial time.

A problem L is NP-complete if:
1) L is NP-hard; and
2) L is in NP.

• One NPC problem is in P \implies P=NP

Open question: is P=NP ?
Satisfiability

**SAT**: is a given n-variable boolean formula (in CNF) satisfiable?

CNF (Conjunctive Normal Form): i.e., product-of-sums
"satisfiable" ⇒ can be made "true"

Ex: \((x+y)(\overline{x} +z)\) is satisfiable

\[(x+z)(\overline{x})(\overline{z})\] is not satisfiable

**3-SAT**: is a given n-var boolean formula (in 3-CNF) satisfiable?

3-CNF: three literals per clause

Ex: \((x_1+x_5+x_7)(x_3+\overline{x}_4+\overline{x}_5)\)
Cook's Theorem

**Thm:** SAT is NP-complete [Cook 1971]

**Pf idea:** given a non-deterministic polynomial-time TM $M$ and input $w$, construct a CNF formula that is satisfiable iff $M$ accepts $w$.

Use variables:

- $q[i,k] \implies$ at step $i$, $M$ is in state $k$
- $h[i,k] \implies$ at step $i$, read-write head scans tape cell $k$
- $s[i,j,k] \implies$ at step $i$, tape cell $j$ contains symbol $\Sigma_k$

$M$ always halts in polynomial time

$\implies$ # of variables is polynomial
Clauses for necessary restrictions:

- At each time i:
  - M is in exactly 1 state
  - r/w head scans exactly 1 cell
  - all cells contain exactly 1 symb

- Time 0 ⇒ initial state
- Time P(n) ⇒ final state
- Transitions from time i to time i+1 obey M's transition function

Resulting formula is satisfiable iff M accepts w.

Thm: 3-SAT is NP-complete

Pf idea: convert each long clause to an equivalent set of short ones:

\[(x+y+z+u+v+w)\]

\[\Rightarrow (x+y+a)(\overline{a} +z+b)(\overline{b} +u+c)(\overline{c} +v+w)\]
Q: is 1-SAT NP-complete?

Q: is 2-SAT NP-complete?
**COLORABILITY**: given a graph $G$ and integer $k$, is $G$ $k$-colorable?

(different colors for adjacent nodes)

Ex:

![Graph Example]

**Thm**: 3-COLORABILITY is NPC

**Proof**: reduction from 3-SAT

$$(x+y+z) \Rightarrow \text{gadget is 3-colorable } \iff x+y+z \text{ is true}$$

![Gadget Diagram]

$$\forall x$$
Ex: \((x+y+z)(\overline{x} + \overline{y} + z)(\overline{x} + y + \overline{z})\)
Ex (cont.): a 3-coloring:

Solution ⇒ x=true, y=false, z=false
**Thm:** 3-COLORABILITY is NPC for graphs with max degree 4.

**Pf:** degree-reduction "gadget":

1. max degree 4
2. 3-colorable but not 2-colorable
3. all corners get same color

"Super"-gadgets:

Use these "fanout" components to reduce node degrees to 4 or less
G is 3-colorable $\iff$ $G'$ is 3-colorable
Q: is 3-COLORABILITY NPC for graphs with max degree 3?
Thm: 3-COLORABILITY is NPC for planar graphs.

Pf: planarity-preserving "gadget":

a) planar and 3-colorable
b) Opposite Corners get same color
c) "independence" of pairs of OC's

Use gadget to avoid edge crossings:
Ex:

G:

G':

G is 3-colorable $\iff$ G' is 3-colorable