# 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

Cormen, Leiserson, Rivest, and Stein, Introduction to Algorithms, Second Edition, McGraw-Hill, 2001.



## Suggested Reading

Polya, How to Solve it, Princeton University Press, 1957.

Preparata and Shamos, Computational Geometry, an Introduction, Springer-Verlag, 1985.

Miyamoto Musashi, Book of Five Rings, Overlook Press, 1974.

*"This book fills a much-needed gap."*
*- Moses Hadas (1900-1966) in a review*

# Grading scheme

Homeworks: 25%

Midterm: 25%

Final: 25%

Project: 25%

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: 210 Olsson 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 Neumann (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" $\wedge$
- "or" $\vee$
- "not" $\neg$
- "xor" $\oplus$
- "nand"
- "nor"
- "implication" $\Rightarrow$
- "equivalence" $\Leftrightarrow$



- "not" $\neg$

  "negation"

Truth table:

| p | $\neg$p |
|---|---|
| T | F |
| F | T |

Ex: let p="today is Monday"

$\neg$p ="today is not Monday"

- “<u>and</u>” $\wedge$

  “conjunction”

Truth table:

| p | q | p∧q |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

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

$(x\geq 0) \wedge (x\leq 10)$



- “<u>or</u>” $\vee$

  “disjunction”

Truth table:

| p | q | p∨q |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

Ex: $(x\geq 7) \vee (x=3)$

$(x=0) \vee (y=0)$



- “<u>xor</u>” $\oplus$

  “exclusive or”

Truth table:

| p | q | p⊕q |
|---|---|---|
| T | T | F |
| T | F | T |
| F | T | T |
| F | F | F |

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

“it is midnight” $\oplus$ “it is sunny”



# <u>Logical Implication</u>

- “<u>implies</u>” $\Rightarrow$

Truth table:

| p | q | p⇒q |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

Ex: $(x\leq 0) \wedge (x\geq 0) \Rightarrow (x=0)$

$1 < x < y \Rightarrow x^3 < y^3$

“today is Sunday” $\Rightarrow$ $1+1=3$

Other interpretations of $p \Rightarrow q$:

- "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" $\Leftrightarrow$

or "if and only if" ("iff")
or "necessary and sufficient"
or "logically equivalent" $\equiv$

Truth table:

| p | q | $p \Leftrightarrow q$ |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | T |

Ex: $p \Leftrightarrow p$

$[(x=0) \vee (y=0)] \Leftrightarrow (xy=0)$

$\min(x,y)=\max(x,y) \Leftrightarrow x=y$



*logically equivalent* ($\Leftrightarrow$) - means "has same truth table"

Ex: $p \Rightarrow q$ is equivalent to $(\neg p) \vee q$

i.e., $p \Rightarrow q \Leftrightarrow (\neg p) \vee q$

| p | q | $p \Rightarrow q$ | $\neg p$ | $\neg p \vee q$ |
|---|---|---|---|---|
| T | T | T | F | T |
| T | F | F | F | F |
| F | T | T | T | T |
| F | F | T | T | T |

Ex: $(p \Leftrightarrow q) \equiv [(p \Rightarrow q) \wedge (q \Rightarrow p)]$

$p \Leftrightarrow q \equiv p \Rightarrow q \wedge q \Rightarrow p$

$(p \Leftrightarrow q) \equiv [(\neg p \vee q) \wedge (\neg q \vee p)]$



Note: $p \Rightarrow q$ is not equivalent to $q \Rightarrow p$

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

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

A: $\neg(p \Rightarrow q) \equiv \neg(\neg p \vee q) \equiv p \wedge \neg q$

| p | q | $\neg q$ | $p \Rightarrow q$ | $\neg(p \Rightarrow q)$ | $p \wedge \neg q$ |
|---|---|---|---|---|---|
| T | T | F | T | F | F |
| T | F | T | F | T | T |
| F | T | F | T | F | F |
| F | F | T | T | F | F |

*"Logic is in the eye of the logician."*
*- Gloria Steinem*

# Example

let p = "it is raining"
let q = "the ground is wet"

$p \Rightarrow q$ : "if it is raining, then the ground is wet"

$\neg q \Rightarrow \neg p$ : "if the ground is not wet, then it is not raining"

$q \Rightarrow p$ : "if the ground is wet, then it is raining"

$\neg(p \Rightarrow q)$ : "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 $p \Rightarrow q$ is $q \Rightarrow p$
*contrapositive* of $p \Rightarrow q$ is $\neg q \Rightarrow \neg p$

Prove: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$



Q: How many distinct 2-variable Boolean functions are there?



# Bit Operations

| $\neg$ | |
|---|---|
| 0 | 1 |
| 1 | 0 |

| $\wedge$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| $\vee$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| $\Rightarrow$ | 0 | 1 |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 1 |

| $\Leftrightarrow$ | 0 | 1 |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 1 |

# Bit Strings

Def: *bit string* - sequence of bits

Boolean functions extend to bit strings (bitwise)

Ex: $\neg\, 0100 = 1011$

$0100 \wedge 1110 = 0100$

$0100 \vee 1110 = 1110$

$0100 \oplus 1110 = 1010$

$0100 \Rightarrow 1110 = 1111$

$0100 \Leftrightarrow 1110 = 0101$



# Proposition types

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

Ex: $p \vee \neg p$ is a tautology

$p \wedge \neg p$ is a contradiction

$p \Rightarrow \neg p$ is a contingency

| p | ¬p | p∨¬p | p∧¬p | p⇒¬p |
|---|---|---|---|---|
| T | F | T | F | F |
| F | T | T | F | T |



# Logic Laws

Identity:

$p \wedge T \Leftrightarrow p$
$p \vee F \Leftrightarrow p$

Domination:

$p \vee T \Leftrightarrow T$
$p \wedge F \Leftrightarrow F$

Idempotent:

$p \vee p \Leftrightarrow p$
$p \wedge p \Leftrightarrow p$



# Logic Laws (cont.)

Double Negation:

$\neg(\neg p) \Leftrightarrow p$

Commutative:

$p \vee q \Leftrightarrow q \vee p$
$p \wedge q \Leftrightarrow q \wedge p$

Associative:

$(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$
$(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$

# Logic Laws (cont.)

Distributive:

$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
$p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$

De Morgan's:

$\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$
$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$

Misc:

$p \vee \neg p \Leftrightarrow T$
$p \wedge \neg p \Leftrightarrow F$
$(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$



# Example

Simplify the following:

$(p \wedge q) \Rightarrow (p \vee q)$



# Predicates

Def: *predicate* - a function or formula involving some variables

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

$P(5)$

$P(1)$

Ex: $Q(x,y,z)$ = "$x^2+y^2=z^2$"

$Q(2,3,4)$

$Q(3,4,5)$



# Quantifiers

- Universal: "for all" $\forall$
  $\forall x\ P(x)$
  $\Leftrightarrow P(x_1) \wedge P(x_2) \wedge P(x_3) \wedge \ldots$
  Ex: $\forall x \quad x < x + 1$
  $\forall x \quad x < x^3$
- Existential: "there exists" $\exists$
  $\exists x\ P(x)$
  $\Leftrightarrow P(x_1) \vee P(x_2) \vee P(x_3) \vee \ldots$
  Ex: $\exists x \quad x = x^2$
  $\exists x \quad x < x - 1$

Combinations:

$\forall x\ \exists y \quad y > x$

# Examples

- $\forall x\ \exists y \quad x+y=0$
- $\exists y\ \forall x \quad x+y=0$
- "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 ):

  $n|j \Leftrightarrow \exists k \in \mathbb{Z}\ n=kj$
- m is prime (denoted P(m)):

  $P(m) \Leftrightarrow [\forall i \in \mathbb{Z}\ (m|i) \Rightarrow (i=m) \vee (i=1)]$
- "there is no largest prime"

  $\forall p\ \exists q \in \mathbb{Z}\ (q>p) \wedge P(q)$

  $\forall p\ \exists q \in \mathbb{Z}\ (q>p) \wedge$
  $[\forall i \in \mathbb{Z}\ (q|i) \Rightarrow (i=q) \vee (i=1)\ ]$

  $\forall p\ \exists q \in \mathbb{Z}\ (q>p) \wedge$
  $[\forall i \in \mathbb{Z}\ \{\exists k \in \mathbb{Z}\ q=ki\} \Rightarrow (i=q) \vee (i=1)]$



# Negation of Quantifiers

Thm: $\neg(\forall x\ P(x)) \Leftrightarrow \exists x\ \neg P(x)$

Ex: ¬ "all men are mortal"
⇔ "there is a man who is not mortal"

---

Thm: $\neg(\exists x\ P(x)) \Leftrightarrow \forall x\ \neg P(x)$

Ex: ¬ "there is a planet with life on it"
⇔ "all planets do not contain life"

---

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

Ex: ¬ "there is a man that exercises every day"
⇔"every man does not exercise some day"

---

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

Ex: ¬ "all things come to an end"
⇔"some thing does not come to any end"



# Quantification Laws

Thm: $\forall x\ (P(x) \wedge Q(x))$
$\Leftrightarrow (\forall x\ P(x)) \wedge (\forall x\ Q(x))$

Thm: $\exists x\ (P(x) \vee Q(x))$
$\Leftrightarrow (\exists x\ P(x)) \vee (\exists x\ Q(x))$

---

Q: Are the following true?

$\exists x\ (P(x) \wedge Q(x))$
$\Leftrightarrow (\exists x\ P(x)) \wedge (\exists x\ Q(x))$

$\forall x\ (P(x) \vee Q(x))$
$\Leftrightarrow (\forall x\ P(x)) \vee (\forall x\ Q(x))$

# More Quantification Laws

- $(\forall x\ Q(x)) \wedge P \Leftrightarrow \forall x\ (Q(x) \wedge P)$
- $(\exists x\ Q(x)) \wedge P \Leftrightarrow \exists x\ (Q(x) \wedge P)$
- $(\forall x\ Q(x)) \vee P \Leftrightarrow \forall x\ (Q(x) \vee P)$
- $(\exists x\ Q(x)) \vee P \Leftrightarrow \exists x\ (Q(x) \vee P)$



# 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
- …

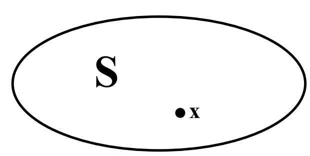


# Sets

Def: *set* - an unordered collection of elements

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

Venn Diagram:




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

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

{0} ≠ {1}

{3, 5} = {3, 5, 3, 3, 5}

- Set <u>construction</u>:
  $|$ or $\ni$ means "such that"

  Ex: $\{k \mid 0<k<4\}$

  $\{k \mid k \text{ is a perfect square}\}$

- Set <u>membership:</u> $\in$ $\notin$

  Ex: $7 \in \{p \mid p \text{ prime}\}$

  $q \notin \{0, 2, 4, 6, ...\}$

- Sets can contain other sets

  Ex: $\{2, \{5\}\}$

  $\{\{\{0\}\}\} \neq \{0\} \neq 0$

  $S = \{1, 2, 3, \{1\}, \{\{2\}\}\}$



# Common Sets

<u>Naturals</u>: $\mathsf{N} = \{1, 2, 3, 4, ...\}$

<u>Integers</u>: $\mathsf{Z} = \{..,-2, -1, 0, 1, 2,..\}$

<u>Rationals</u>: $Q = \{ \frac{a}{b} \mid a,b \in Z, b \neq 0\}$

<u>Reals</u>: $\Re = \{x \mid x \text{ a real } \#\}$

<u>Empty set</u>: $\emptyset = \{\}$

$\mathsf{Z}^{+}$ = non-negative integers

$\Re^{-}$ = 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"

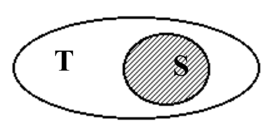
$(1,2) \neq (2,1)$ "2-tuple"



# Subsets

- <u>Subset</u> notation: $\subseteq$

  $S \subseteq T \Leftrightarrow (x \in S \Rightarrow x \in T)$




- <u>Proper subset</u>: $\subset$

  $S \subset T \Leftrightarrow ((S \subseteq T) \wedge (S \neq T))$

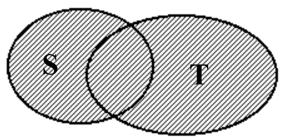
  $S = T \Leftrightarrow ((T \subseteq S) \wedge (S \subseteq T))$

  $\forall S \ \emptyset \subseteq S$

  $\forall S \ S \subseteq S$

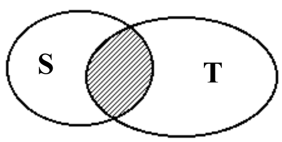
- <u>Union</u>: $\cup$

  $S \cup T = \{x \mid x \in S \vee x \in T\}$




- <u>Intersection</u>: $\cap$

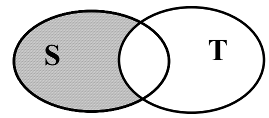
  $S \cap T = \{x \mid x \in S \wedge x \in T\}$





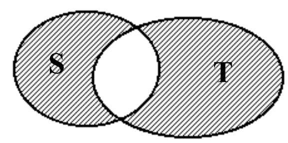

- Set <u>difference</u>: $S - T$

  $S - T = \{x \mid x \in S \wedge x \notin T\}$




- <u>Symmetric difference</u>: $S \oplus T$

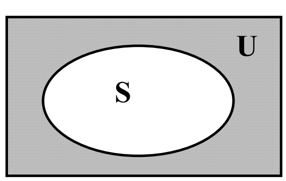
$$S \oplus T = \{x \mid x \in S \oplus x \in T\}$$
$$= S \cup T - S \cap T$$



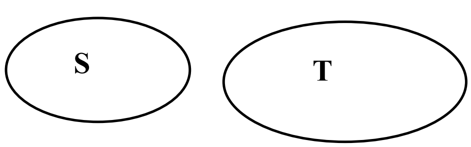



- Universal set: U (everything)
- Set <u>complement</u>: S' or $\overline{S}$

  $S' = \{x \mid x \notin S\} = U - S$




- <u>Disjoint</u> sets: $S \cap T = \emptyset$




$S - T = S \cap T'$

$S - S = \emptyset$



# <u>Examples</u>

$N \cup Z \cup Q \cup \Re = \Re$

$N \subset Z \subset Q \subset \Re$

$\forall x \in \Re \;\; x \leq x^2+1$

$\forall x,y \in Q \; \min(x,y)=\max(x,y) \Leftrightarrow x=y$

$\Re^+ \cup \Re^- = \Re$

$\Re^+ \cap \Re^- = \{0\}$

# 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:

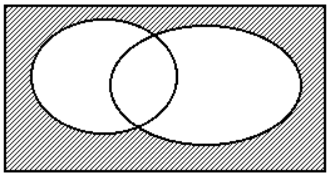
$S \cup (S \cap T) = S$

$S \cap (S \cup T) = S$

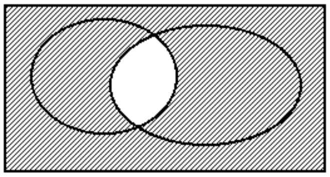


# DeMorgan's Laws

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



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



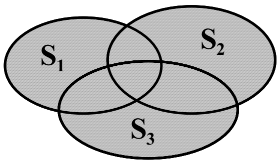
Boolean logic version:

$(X \wedge Y)' = X' \vee Y'$
$(X \vee Y)' = X' \wedge Y'$

# Generalized $\cup$ and $\cap$

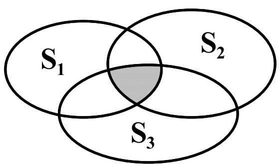
- $\bigcup_{1 \le i \le n} S_i = S_1 \cup S_2 \cup S_3 \cup \ldots \cup S_n$

  $= \{x \mid \exists i\ 1 \le i \le n \ni x \in S_i\}$




- $\bigcap_{1 \le i \le n} S_i = S_1 \cap S_2 \cap S_3 \cap \ldots \cap S_n$

  $= \{x \mid \forall i\ 1 \le i \le n \Rightarrow x \in S_i\}$






# Set Representation

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

Ex: $S = \{x_1, \quad x_3, \quad\quad x_n\}$
bits: 1 0 1 0 ... 0 0 1

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

- "or" yields union:

| | | |
|---|---|---|
| | 1010000...01 | $\{x_1, x_3, x_n\}$ |
| $\vee$ | <u>0111000...00</u> | $\{x_2, x_3, x_4\}$ |
| | 1111000...01 | $\{x_1, x_2, x_3, x_4, x_n\}$ |

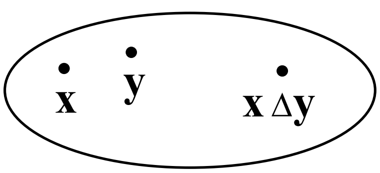
- "and" yields intersection:

| | | |
|---|---|---|
| | 1010000...01 | $\{x_1, x_3, x_n\}$ |
| $\wedge$ | <u>0111000...00</u> | $\{x_2, x_3, x_4\}$ |
| | 0010000...00 | $\{x_3\}$ |



- Set <u>closure</u>: WRT operation $\Delta$

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




- Ex: $\Re$ is closed under addition since $x,y \in \Re \Rightarrow x+y \in \Re$

# Abbreviations

- WRT "with respect to"
- WLOG "without loss of generality"

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



# Cartesian Product

- <u>Ordered n-tuple</u>: element sequence

  Ex: (2,3,5,7) is a 4-tuple

- <u>Tuple equality</u>:

  $(a,b)=(x,y) \Leftrightarrow (a=x) \wedge (b=y)$
  Generally: $(a_i)=(x_i) \Leftrightarrow \forall i\ \ a_i=x_i$

- <u>Cross-product</u>: 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:

  $S_1 \times S_2 \times \ldots \times S_n$
  $= \{(x_1,\ldots,x_n) \mid x_i \in S_i,\ 1 \le i \le n\}$

  $T^i = T \times T^{i-1}$
  $T^1 = T$

- Euclidean plane = $\Re \times \Re = \Re^2$
- Euclidean space = $\Re \times \Re \times \Re = \Re^3$
- Russel's paradox: set of all sets that do not contain themselves:

  $\{S \mid S \notin S\}$

  Q: Does S contain itself??

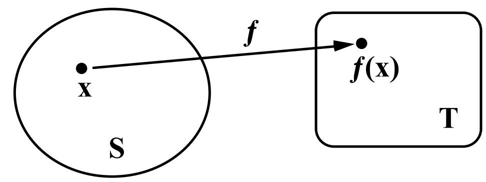


# Functions

- Function: mapping $f{:}S \to T$

  Domain S

  Range T




- k-ary: has k "arguments"
- Predicate: with range = {true, false}



# Function Types

- One-to-one function: "1-1"
  $a,b \in S \wedge a \neq b \Rightarrow f(a) \neq f(b)$

  Ex: $f: \Re \to \Re$, $f(x)=2x$ is 1-1
  $g(x)=x^2$ is not 1-1

- Onto function:

  $\forall\ t \in T\ \exists\ s \in S \ni f(s)=t$

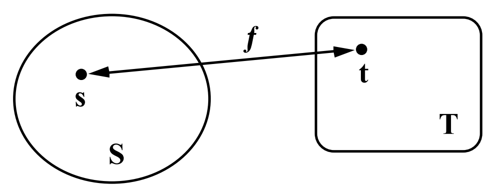
  Ex: $f: Z \to Z$, $f(x)=13-x$ is onto
  $g(x)=x^2$ is not onto



# 1-to-1 Correspondence

- 1-to-1 correspondence: $f{:}S \leftrightarrow T$

  $f$ is both 1-1 and onto




Ex: $f: \Re \leftrightarrow \Re \ni f(x)=x$ (identity)

$h: N \leftrightarrow Z \ni h(x)= \frac{x-1}{2}$, x odd,
$\frac{-x}{2}$, x even.

- Inverse function:

$f$:S→T $\quad f^{-1}$:T→S

$f^{-1}$(t)=s if $f$(s)=t

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

- Function composition:

β:S→T, α:T→V
⇒ (α • β)(x)=α(β(x))
(α • β):S→V

Ex: β(x)=x+1 $\quad$ α(x)=$x^2$
(α • β)(x)= $x^2 + 2x + 1$



Thm: $(f \bullet f^{-1})(x) = (f^{-1} \bullet f)(x) = x$



# Set Cardinality

- Cardinality: |S| = #elements in S

Ex: |{a,b,c}|=3

|{p | p prime < 9}| = 4

|Ø|=0

|{{1,2,3,4,5}}| = ?

- Powerset: $2^S$ = set of all subsets

$2^S$ ={T | T ⊆ S}

Ex: $2^{\{a,b\}}$ = {{},{a},{b},{a,b}}

Q: What is $2^{Ø}$ ?



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 <u>at least as large</u> as T:
  $|S| \geq |T| \Rightarrow \exists\, f: S \rightarrow T$, $f$ onto
  i.e., "S covers T"
  Ex: $r: \Re \rightarrow \mathsf{Z}$, $r(x) = \text{round}(x)$
  $\Rightarrow |\Re| \geq |Z|$
- S and T have <u>same cardinality</u>:
  $|S| = |T| \Rightarrow |S| \geq |T| \wedge |T| \geq |S|$
  or
  $\exists$ 1-1 correspondence $S \leftrightarrow T$
- Generalizes finite cardinality:
  $\{1, 2, 3, 4, 5\} \geq \{a, b, c\}$



## Infinite Sets

- <u>Infinite set</u>: $|S| > k\ \forall k \in \mathsf{Z}$
  or
  $\exists$ 1-1 corres. $f: S \leftrightarrow T$, $S \subset T$
  Ex: $\{p \mid p \text{ prime}\}$, $\Re$
- <u>Countable set</u>: $|S| \leq |\mathsf{N}|$
  Ex: $\emptyset$, $\{p \mid p \text{ prime}\}$, N, Z
- S is <u>strictly smaller</u> than T:
  $|S| < |T| \Rightarrow |S| \leq |T| \wedge |S| \neq |T|$
- <u>Uncountable set</u>: $|\mathsf{N}| < |S|$
  Ex: $|\mathsf{N}| < \Re$
  $|\mathsf{N}| < [0,1] = \{x \mid x \in \Re,\ 0 \leq x \leq 1\}$



<u>Thm</u>: $\exists$ 1-1 correspondence $\mathsf{Q} \leftrightarrow \mathsf{N}$
<u>Pf</u> (<u>dove-tailing</u>):

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | |
| 6 | $\frac{1}{6}$ | $\frac{2}{6}$ | $\frac{3}{6}$ | $\frac{4}{6}$ | $\frac{5}{6}$ | $\frac{6}{6}$ | … |
| 5 | $\frac{1}{5}$ | $\frac{2}{5}$ | $\frac{3}{5}$ | $\frac{4}{5}$ | $\frac{5}{5}$ | $\frac{6}{5}$ | … |
| 4 | $\frac{1}{4}$ | $\frac{2}{4}$ | $\frac{3}{4}$ | $\frac{4}{4}$ | $\frac{5}{4}$ | $\frac{6}{4}$ | … |
| 3 | $\frac{1}{3}$ | $\frac{2}{3}$ | $\frac{3}{3}$ | $\frac{4}{3}$ | $\frac{5}{3}$ | $\frac{6}{3}$ | … |
| 2 | $\frac{1}{2}$ | $\frac{2}{2}$ | $\frac{3}{2}$ | $\frac{4}{2}$ | $\frac{5}{2}$ | $\frac{6}{2}$ | … |
| 1 | $\frac{1}{1}$ | $\frac{2}{1}$ | $\frac{3}{1}$ | $\frac{4}{1}$ | $\frac{5}{1}$ | $\frac{6}{1}$ | … |
| | 1 | 2 | 3 | 4 | 5 | 6 | |



<u>Thm</u>: $|\Re| > |\mathsf{N}|$
<u>Pf</u> (<u>diagonalization</u>):

Assume $\exists$ 1-1 corres. $f: \Re \leftrightarrow \mathsf{N}$
Construct $x \in \Re$:

```
f(1)=2.718281828...   → 8
f(2)=1.414213562...   → 2
f(3)=1.618033989...   → 9
```

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

$\Rightarrow f$ not a 1-1 correspondence

$\Rightarrow$ contradiction

$\Rightarrow \Re$ is uncountable

Q: Is $|2^{Z}| = |\Re|$ ?



Q: Is $|\Re| > |[0,1]|$ ?



<u>Thm</u>: any set is "smaller" than its powerset.

$$|S| < |2^{S}|$$



# <u>Infinities</u>

- $|N| = \aleph_0$
- $|\Re| = \aleph_1$
- $\aleph_0 < \aleph_1 = 2^{\aleph_0}$
- "<u>Continuum Hypothesis</u>"

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

  <u>Independent of the axioms</u>! [Cohen, 1966]
- <u>Axiom of choice</u> [Godel 1938]
- <u>Parallel postulate</u>

## Infinity Hierarchy

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

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

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

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

- 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>"
.
.

$\Rightarrow$ # 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), ...}

$\Rightarrow$ { 2, 3, 5, ...} $\in 2^N$

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

$|2^N|$ is uncountable (why?)

$\Rightarrow$ # functions is uncountable!



Thm: most functions are uncomputable!

Pf: # algorithms is countable
# functions is not countable

$\Rightarrow \exists$ more functions than algorithms / programs!

$\Rightarrow$ some functions do not have algorithms!

---

Ex: The halting problem

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

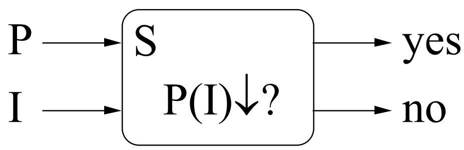
Def: H(P,I) = 1 if P halts on I
0 otherwise

# The Halting Problem

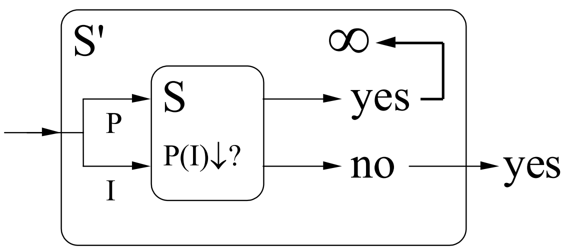
H: Given a program P and input I, does P halt on I? i.e., does $P(I)\downarrow$ ?

Thm: H is uncomputable

Pf: Assume subroutine S solves H.

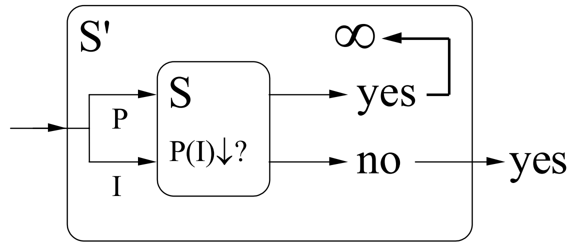



Construct:






Analyze:




$S'(S')\downarrow \Rightarrow S'(S')\uparrow$

$S'(S')\uparrow \Rightarrow S'(S')\downarrow$

so, $S'(S')\uparrow \Leftrightarrow S'(S')\downarrow$

a contradiction!

$\Rightarrow$ S does not correctly compute H

But S was an arbitrary subroutine, so

$\Rightarrow$H 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|

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

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 $p(x_i)$

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

# Conditional Probability

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

$p(E \cap F) = p(F)\, p(E \mid F)$

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)}
= both are male

$p(E \cap F) = p(F)\, p(E \mid F)$

$p(E \mid F) = p(E \cap F) / p(F)$
$= (1/4) / (3/4) = 1/3$



# 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: $x \heartsuit x\ \forall x$

Symmetric: $x \heartsuit y \Rightarrow y \heartsuit x$

Transitive: $x \heartsuit y \wedge y \heartsuit z \Rightarrow x \heartsuit z$

Antisymmetric: $x \heartsuit y \Rightarrow \neg(y \heartsuit x)$

Ex: $\leq$ is reflexive
transitive
not symmetric

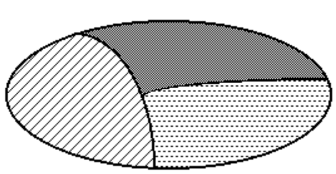


# Equivalence Relations

Def: reflexive, symmetric, & transitive

Ex: standard equality "="
$x=x$
$x=y \Rightarrow y=x$
$x=y \wedge y=z \Rightarrow x=z$

Partition - disjoint equivalence classes:





# Closures

- Transitive closure of ♥: TC
smallest superset of ♥ satisfying
$x \heartsuit y \wedge y \heartsuit z \Rightarrow x \heartsuit z$

Ex: "predecessor"
$\{(x-1,x) \mid x \in \mathbb{Z}\}$
TC(predecessor) is "<" relation

- Symmetric closure of ♥:
smallest superset of ♥ 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

$f(n) = O(g(n)) \Leftrightarrow \exists\ c,k > 0$
$\ni |f(n)| \le c\cdot|g(n)|\ \ \forall\ n>k$

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

$f(n)=\Omega(g(n)) \Leftrightarrow g(n)=O(f(n))$

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

$f(n) = \Theta(g(n)) \Leftrightarrow$
$f(n)=O(g(n))\ \wedge\ g(n)=O(f(n))$

"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

$f(n) = o(g(n)) \Leftrightarrow$

$f(n)=O(g(n)) \wedge f(n)\neq\Omega(g(n))$

$\lim_{n\to\infty} f(n)/g(n) = 0$

"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 $f_2(n)=O(g_2(n))$

Thm: $f_1(n) + f_2(n) = O(\max(g_1(n),g_2(n)))$

Thm: $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$

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:

$f(n)=O(n^{c-\varepsilon}) \Rightarrow T(n)=\Theta(n^c)$

$f(n)=\Theta(n^c) \Rightarrow T(n)=\Theta(n^c \log n)$

$f(n)=\Omega(n^{c+\varepsilon}) \wedge a \cdot f(n/b) \leq d \cdot f(n)$
$\forall\ d<1, n>n_0 \Rightarrow T(n)=\Theta(f(n))$

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 $\lceil \frac{M}{N} \rceil$ objects.

- Useful in proofs & analyses

# Stirling's Formula

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

$$n! = \sqrt{2\Pi n} \cdot \left(\frac{n}{e}\right)^n \cdot \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

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

$\underline{\log(n!) = O(n \log n)}$

- 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

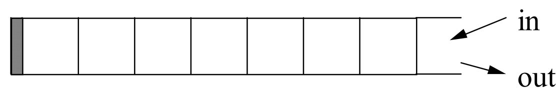
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | . . . |

- 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

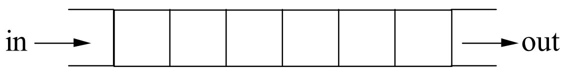
- LIFO (last-in first-out)




- 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)

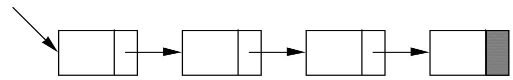



- Operations: push/pop (O(1) each)
- Can not access "middle"
- Analogy: line at your Bank
- Applications:
  - Scheduling
  - Operating systems
  - Simulations
  - Networks

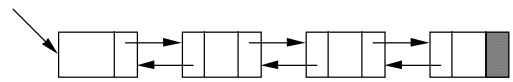


# Linked Lists

- Successor pointers



- Types:
  - Singly linked
  - Doubly linked
  - Circular

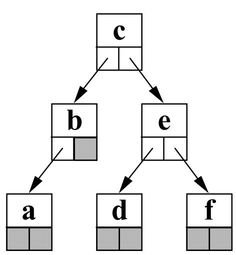


- Operations:
  - Add: O(1) time
  - Search: O(n) time
  - Delete: O(1) time (if known)



# Trees

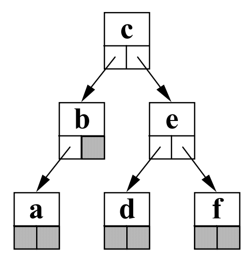
- Parent/children pointers




- 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

  ⇒ c b a e d f

---

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

  ⇒ a b d f e c

---

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

  ⇒ 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 <u>improved</u> to O(log n)?



# Sorting

| |
|---|
| Fact: almost half of <u>all</u> CPU cycles are spent on sorting!! |

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

<u>Thm</u>: Sorting takes $\Omega(n \log n)$ time
<u>Pf</u>: n! different permutations

$\Rightarrow$decision tree has n! leaves

$\Rightarrow$tree height is: $\log(n!)$
$> \log((n/e)^n)$
$= \Omega(n \log n)$



# 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)
```

$\Rightarrow O(n \log n)$ 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:

```
MergeSort(X,i,j)
    if i<j then  m=⌊(i+j)/2⌋
                 MergeSort(X,i,m)
                 MergeSort(X,m+1,j)
                 Merge(X,i,m,j)
```

$$T(n) = 2\,T(n/2) + n$$

$\Rightarrow \Theta(n \log n)$ time

- Heap Sort:

```
    InitHeap
    For i=1 to n HeapInsert(X(i))
    For i=1 to n    M=HeapMax
                    Print(M)
                    HeapDelete(M)
```

$\Rightarrow \Theta(n \log n)$ 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
```

$\Rightarrow \Theta(n)$ time (worst-case)

- Radix Sort:

Assumes d digits in range 1..k

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

$\Rightarrow O(dn+kd)$ time (worst-case)

- Bucket Sort:

Assumes uniform inputs in range 0..1

For i=1 to n
  Insert X[i] into Bucket $\lfloor n \cdot X[i] \rfloor$
For i=1 to n Sort Bucket i
Concat contents of Buckets 1 thru n

$\Rightarrow$O(n) time (expected)
  O(Sort) time (worst)



# Order Statistics

- Exact comparison count
- Minimum element

k=X[1]
For i=2 to n
  If X[i]<k Then k = X[i]

$\Rightarrow$n-1 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

$\Rightarrow$ n/2+n/2+n/2 =3n/2 comparisons

Thm: Min&Max require 3n/2 comparisons.
Pf: Represent known info by four sets:

| | Unknown | Not Min | Not Max | Neither |
|---|---|---|---|---|
| | A | B | C | D |
| Initial: | n | 0 | 0 | 0 |
| Final: | 0 | 1 | 1 | n-2 |

Track movement of elements between sets.



Effect of comparisons:

| Origin | Target < \| > |
|---|---|
| A&A | C&B \| B&C (1) |
| A&B | C&B \| B&D |
| A&C | C&D \| B&C |
| A&D | C&D \| B&D |
| B&B | D&B \| B&D (2) |
| B&C | D&D \| B&C |
| B&D | D&D \| B&D |
| C&C | C&D \| D&C (3) |
| C&D | C&D \| D&D |
| D&D | D&D \| D&D |

- 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

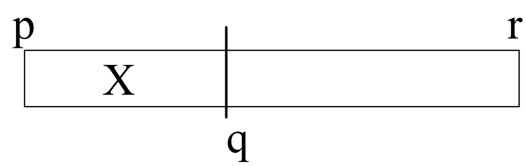
Problem: Find Max and next-to-Max using least # of comparisons.



# 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=RandomPartition(X,p,r)
   k=q-p+1
   If i ≤ k Then Return(Select(X,p,q,i))
         Else Return(Select(X,q+1,r,i-k))
```



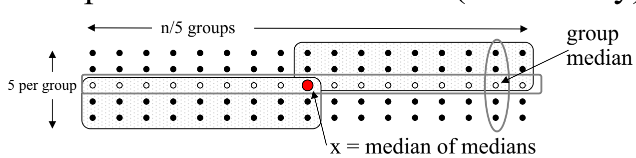

$\Rightarrow$ O(n) time (ave-case)



# 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 ≥ 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 $E \subseteq V \times V$

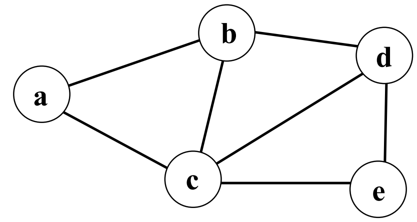
Pictorially: nodes & lines



# Undirected Graphs

Def: edges have no direction

• Example of undirected graph:



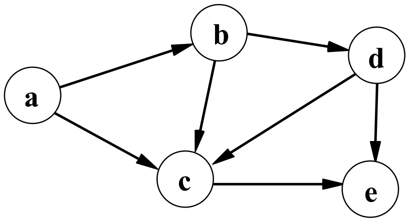

V={a,b,c,d,e}
E={(c,a),(c,b),(c,d),(c,e),
(a,b),(b,d),(d,e)}



# Directed Graphs

Def: edges have direction

• Example of directed graph:



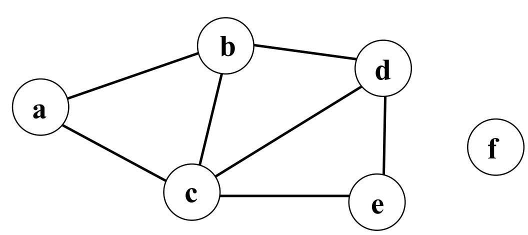

V={a,b,c,d,e}
E={(a,b),(a,c),(b,c),(b,d),
(d,c),(d,e),(c,e)}



# Graph Terminology

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

• node ≡ vertex
• edge ≡ arc




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

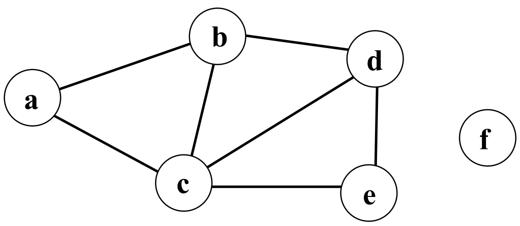
Ex: a & b are neighbors
a & e are not neighbors

# Undirected Node Degree

Degree in <u>undirected</u> graphs:

<u>Degree</u>(v) = # of <u>adjacent</u> (<u>incident</u>) edges to vertex v in G

Ex: deg(c)=4 deg(f)=0



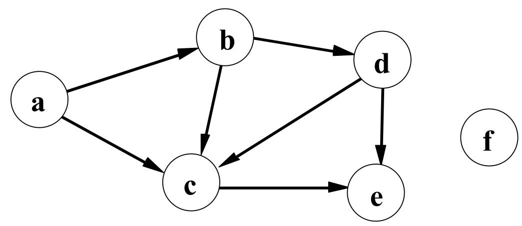



# Directed Node Degree

Degree in <u>directed</u> graphs:

<u>In-degree</u>(v) = # of <u>incoming</u> edges
<u>Out-degree</u>(v) = # of <u>outgoing</u> edges

Ex: in-deg(c)=3 out-deg(c)=1
in-deg(f)=0 out-deg(f)=0

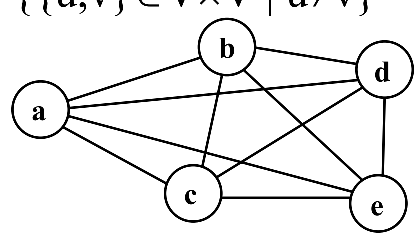





Q: Show that at any party there is an even number of people who shook hands an odd number of times.

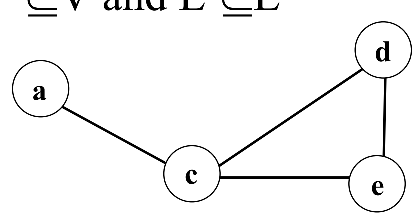


<u>Complete</u> graph $K_n$ contains <u>all</u> edges
i.e., $E = \{\{u,v\} \in V \times V \mid u \neq v\}$




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

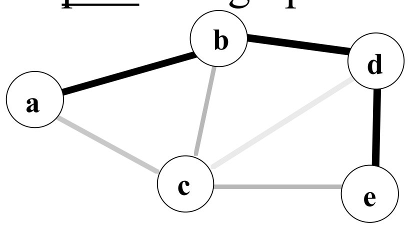
<u>Subgraph</u> of G is G'=(V',E')
where $V' \subseteq V$ and $E' \subseteq E$



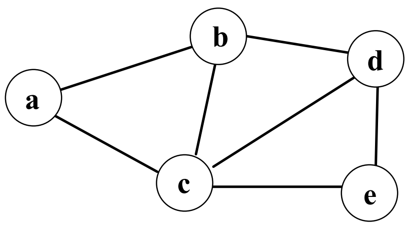

Q: Give a (non-trivial) lower bound on the number of graphs over n vertices.

# Paths in Graphs

Undirected path in a graph:

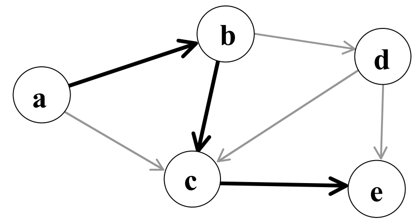



A graph is connected iff there is a path between any pair of nodes:



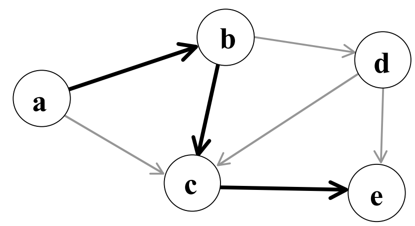



Directed path in a graph:



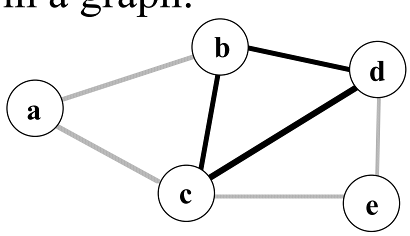

Graph is strongly connected iff there is a directed path between any node pair:

Ex: connected but not strongly:



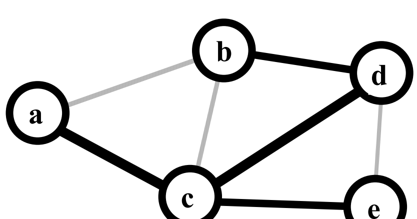



A cycle in a graph:




A tree is an acyclic graph.

Tree T=(V',E') spans G=(V,E) if T is a connected subgraph with V'=V



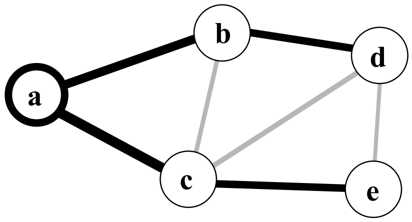



Q: How many edges are there in a tree over n vertices?

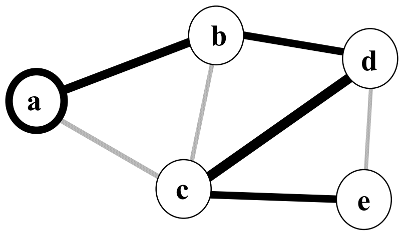
Q: Is the # of distinct spanning trees in a graph G always polynomial in |G|?

# Graph Traversals

Breadth-first search:




Depth-first search:



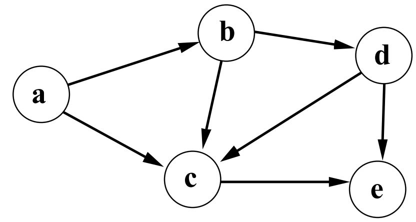

O(E+V) time for either BFS or DFS

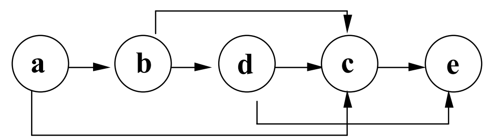
Yields a spanning tree for the graph



# Topological Sort

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



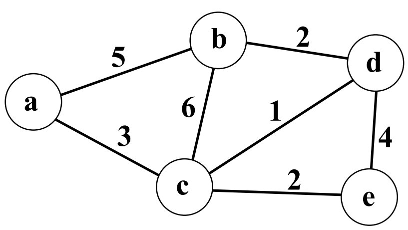

Can be done in O(E+V) time

Application: scheduling w/constraints



# Weighted Graphs

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




Weights can model many things:

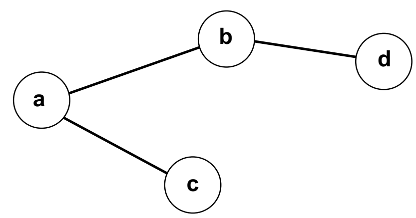
- Distances / lengths
- Speed / time
- Costs

Cost(G) = sum of edge costs

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



# Graph Representation



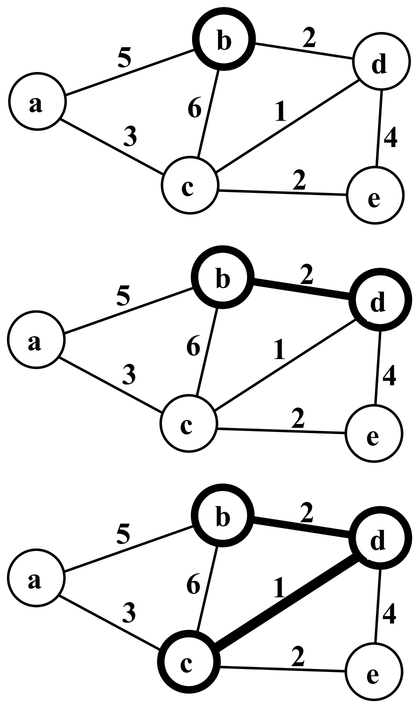

Adjacency list:

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

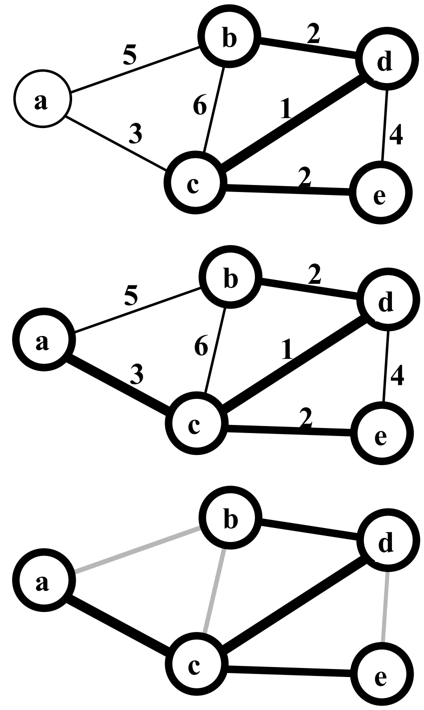
Adjacency matrix:

| | a | b | c | d |
|---|---|---|---|---|
| a | 0 | 1 | 1 | 0 |
| b | 1 | 0 | 0 | 1 |
| c | 1 | 0 | 0 | 0 |
| d | 0 | 1 | 0 | 0 |

# Minimum Spanning Trees







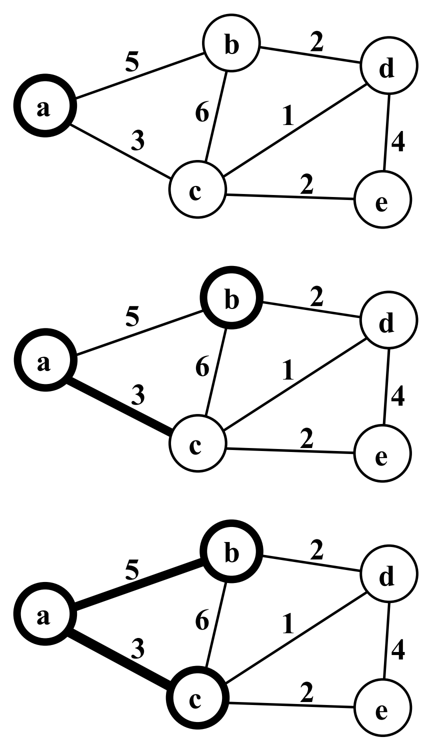

# Prim's MST Algorithm

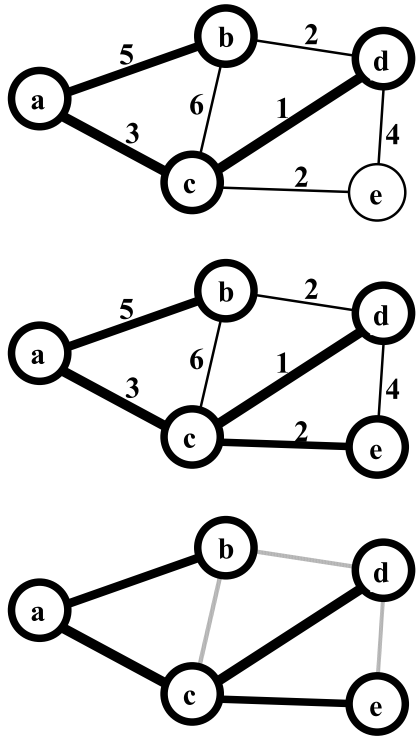
T = $v_0$
**Until** T spans all nodes **do**
  **Select** nodes $x \in T$, $y \notin T$
    w/min cost(x,y)
  **Add** edge (x,y) to T
**Return** T

- Time complexity: O(E log E)
- Kruskal: O(E log V)
- Fibonacci heaps: O(E+VlogV)



# Shortest Paths Trees








# Dijkstra's Single-Source Shortest paths Algorithm

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

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



# Cost-Radius Tradeoffs

Cong, Kahng, Robins, Sarrafzadeh, and Wong, Provably Good Performance-Driven Global Routing, IEEE Transactions on Computer-Aided Design, Vol 11, No. 6, June 1992, pp. 739-752.

**Signal delay $\uparrow \Rightarrow$ Performance $\downarrow$**

- Source $\rightarrow$ sink pathlength $\propto$ delay

  $\Rightarrow$ Avoid long paths

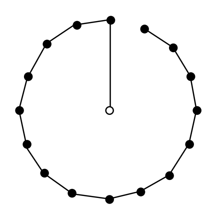
- Capacitive delay / building cost

  $\Rightarrow$ Minimize total wirelength

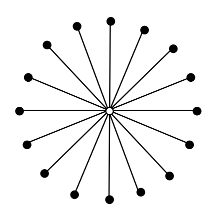


# Possible Trees

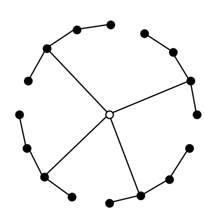
MST:



SPT:



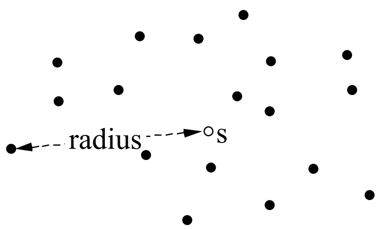
?



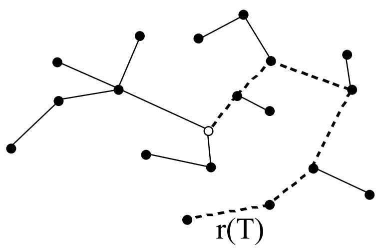
# Definitions

Input: pointset with distinguished source

***ptset radius R***: max source-sink dist




***tree radius***: max source-sink pathlength



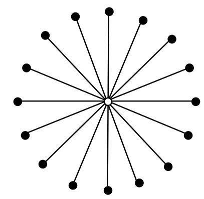



# Problem Formulation

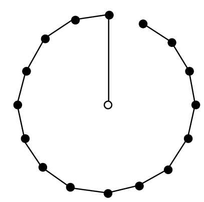
Given a pointset P, $\varepsilon \geq 0$, find min-cost tree T with $r(T) \leq (1+\varepsilon) \cdot R$

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

$\varepsilon = 0 \Rightarrow$ "Shortest Path Tree"

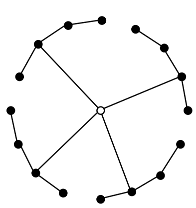


$\varepsilon = \infty \Rightarrow$ Minimum Spanning Tree





Arbitrary $\varepsilon \Rightarrow$ hybrid construction



- Unifies Prim and Dijkstra!

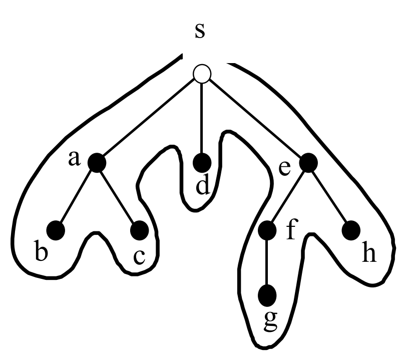


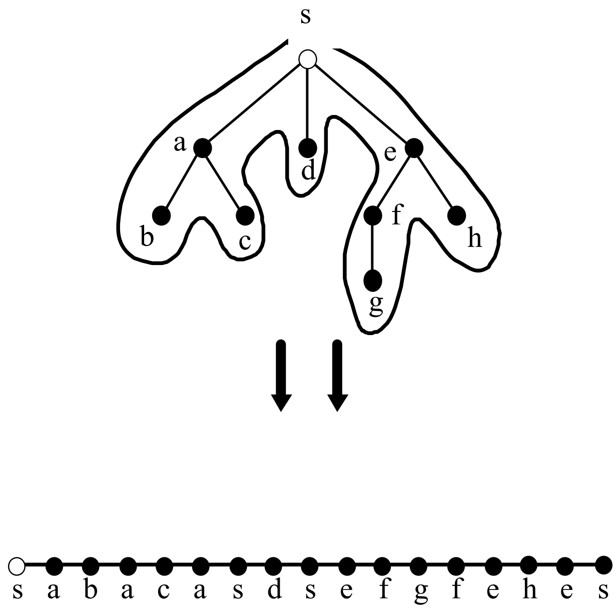
# Bounded Radius MSTs

**Goal**: cost $\approx$ cost(MST)

radius $\approx$ r(SPT)

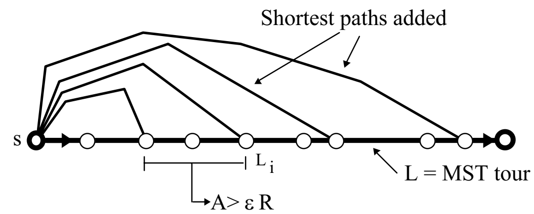
- **Let** Q = MST
- **Let L be** tour of MST:





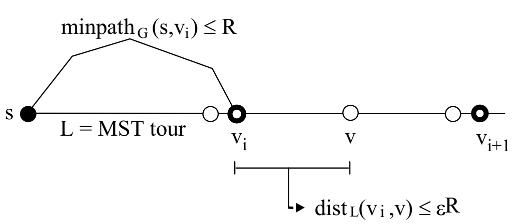



- **Traverse** L
- A = running total of edge costs
- **If** $A > \varepsilon \cdot R$ **Then** $A = 0$
  $Q = Q \cup \text{minpath}_G(s, L_i)$




- Final **routing** tree is $SPT_Q$



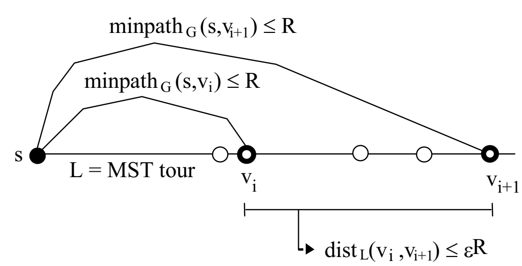



$$\text{dist}_T(s,v) \le \text{dist}_G(s,v_i) + \text{dist}_L(v_i,v)$$

$$\le R + \varepsilon \cdot R = (1+\varepsilon) \cdot R$$

$$\Rightarrow r(T) \le (1+\varepsilon) \cdot R$$






$$\text{cost}(T) \le \text{cost}(MST_G) + \frac{\text{cost}(L)}{\varepsilon \cdot R} \cdot R$$

$$= \text{cost}(MST_G) + \frac{2 \cdot \text{cost}(MST_G)}{\varepsilon}$$

$$= \left(1 + \frac{2}{\varepsilon}\right) \cdot \text{cost}(MST_G)$$

$$\Rightarrow \text{cost}(T) \le \left(1 + \frac{2}{\varepsilon}\right) \cdot \text{cost}(MST_G)$$

# Bounded Radius MST Algorithm

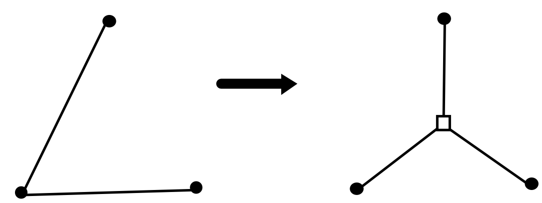
**Compute** $MST_G$ and $SPT_G$
E' = edges of $MST_G$
Q = (V,E')
L = depth-first tour of $MST_G$
A = 0
**For** i = 2 to |L|
  A = A + cost($L_{i-1}$, $L_i$)
  **If** A > $\varepsilon \cdot R$ **Then**
    E' = E' $\cup$ minpath$_G$(s, $L_i$)
    A=0
T = $SPT_Q$

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

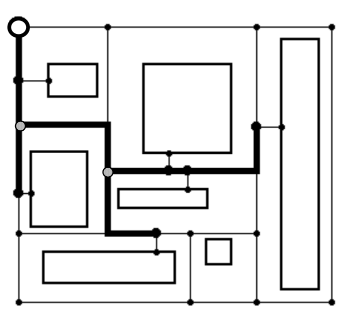
Output: T = routing tree with
$cost(T) \le (1+\frac{2}{\varepsilon}) \cdot cost(MST_G)$
$r(T) \le (1+\varepsilon) \cdot R$



# Steiner Trees







## Bounded Radius Steiner Trees

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

- NP-complete



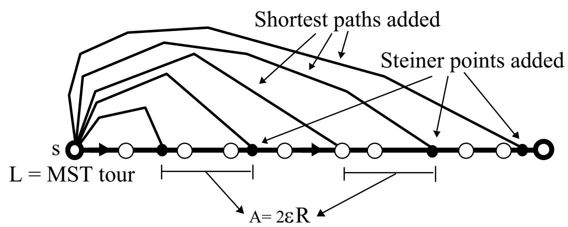
## Bounded Radius Steiner Trees

- Can use *any* low-cost spanning tree
- Use [KMB, 1981] to span N (cost $\le 2 \cdot$opt)
- Run previous algorithm

$$\Rightarrow cost(T) \le 2 \cdot (1+\frac{2}{\varepsilon}) \cdot opt$$

# Geometry Helps

- Add Steiner points when A = 2ε·R

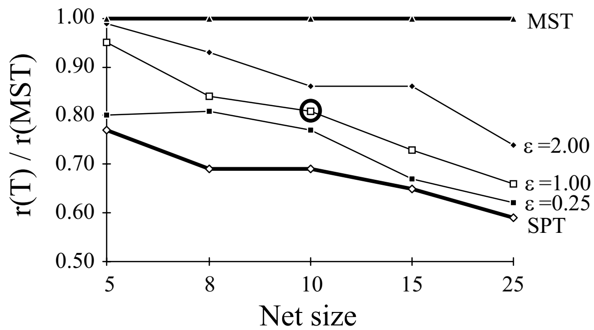


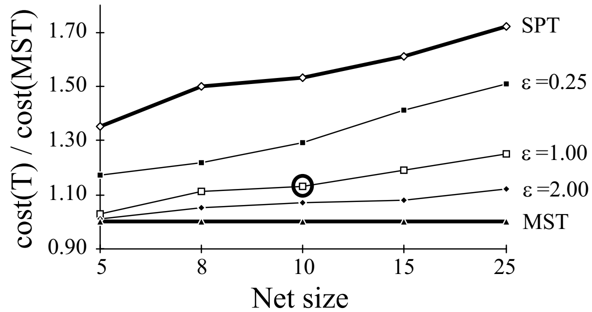

- Use bounds on MST/Steiner ratio

| Tree type | Graph type | Radius bound | Cost bound |
|---|---|---|---|
| spanning | arbitrary | $(1+\varepsilon)\cdot R$ | $(1+2/\varepsilon)\cdot MST$ |
| Steiner | arbitrary | $(1+\varepsilon)\cdot R$ | $2\cdot(1+2/\varepsilon)\cdot opt$ |
| Steiner | Manhattan | $(1+\varepsilon)\cdot R$ | $\frac{3}{2}(1+1/\varepsilon)\cdot opt$ |
| Steiner | Euclidean | $(1+\varepsilon)\cdot R$ | $\frac{2}{\sqrt{3}}\cdot(1+1/\varepsilon)\cdot opt$ |



# Experimental Results






# NP-Completeness

- Tractability
- Polynomial time
- Computation vs. verification
- Non-determinism
- Encodings
- Transformation & reducibilities
- P vs. NP
- "completeness"



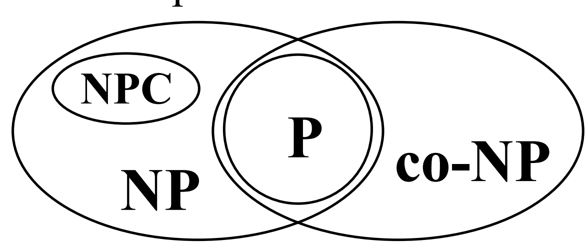
A problem L is NP-hard if:

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

M always halts in polynomial time
$\Rightarrow$ # 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 $\Rightarrow$ initial state
- Time P(n) $\Rightarrow$ 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:

$(x+y+z+u+v+w)$

$\Rightarrow(x+y+a)(\overline{a}+z+b)(\overline{b}+u+c)(\overline{c}+v+w)$

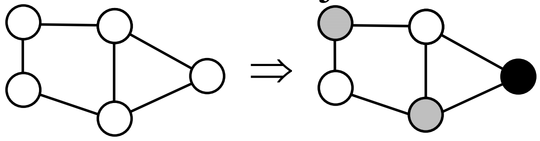


Q: is 1-SAT NP-complete?

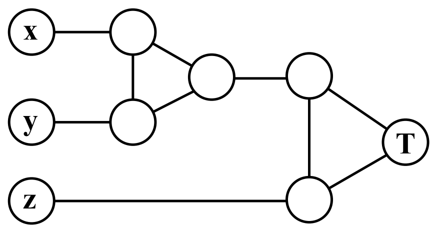
Q: is 2-SAT NP-complete?

COLORABILITY: given a graph G and integer k, is G k-colorable?

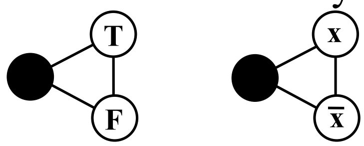
(different colors for adjacent nodes)

Ex: 

Thm: 3-COLORABILITY is NPC
Proof: reduction from 3-SAT

(x+y+z) ⇒ 

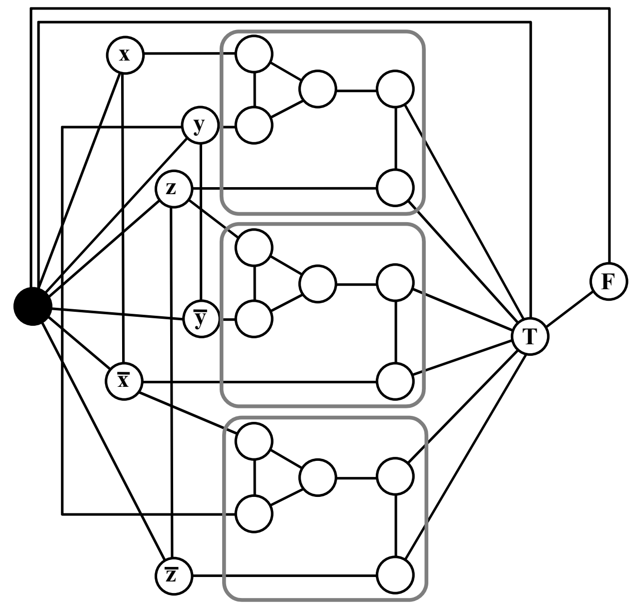

gadget is 3-colorable ⇔ x+y+z is true



$\forall x$

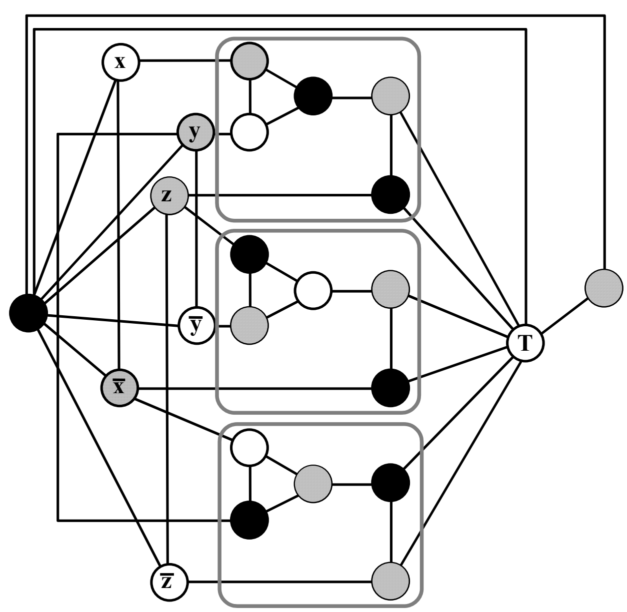


Ex: $(x+y+z)(\overline{x}+\overline{y}+z)(\overline{x}+y+\overline{z})$






Ex (cont.): a 3-coloring:

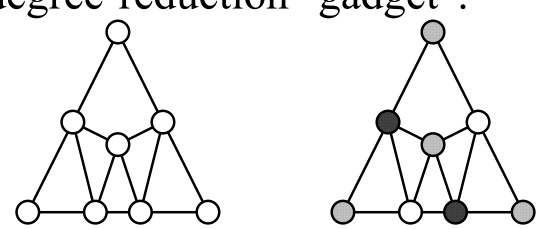



Solution ⇒ x=true, y=false, z=false



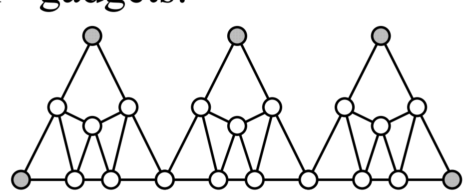
Thm: 3-COLORABILITY is NPC for graphs with max degree 4.

Pf: degree-reduction "gadget":



a) max degree 4
b) 3-colorable but not 2-colorable
c) all corners get same color

"Super"-gadgets:

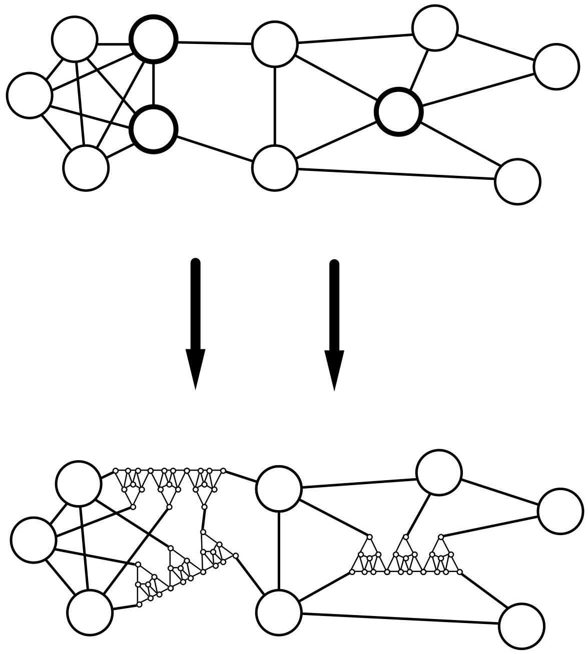


Use these "fanout" components to reduce node degrees to 4 or less

Ex:

G:

G':



G is 3-colorable ⇔ G' is 3-colorable

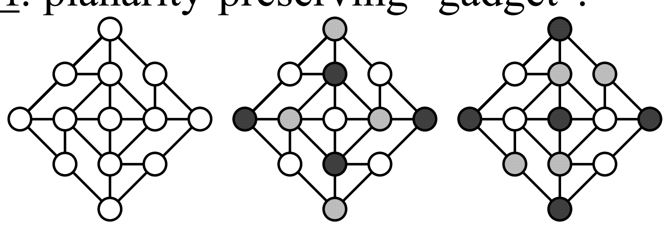


Q: is 3-COLORABILITY NPC for graphs with max degree 3?



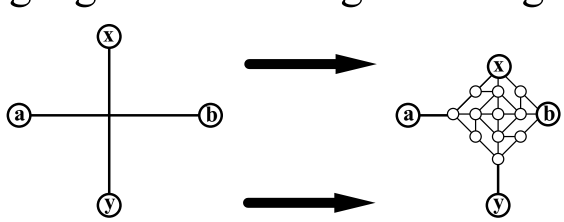
<u>Thm</u>: 3-COLORABILITY is NPC for planar graphs.

<u>Pf</u>: 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:

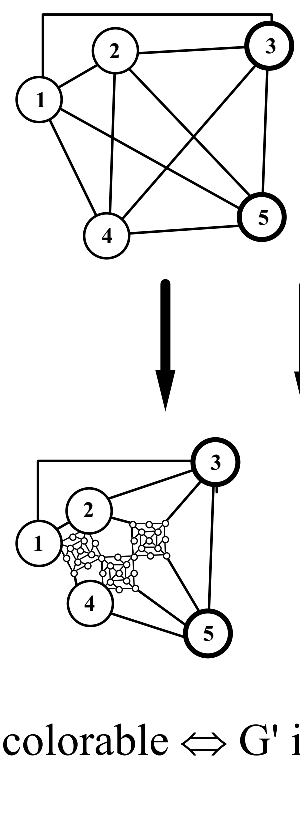





Ex:

G:

G':




G is 3-colorable ⇔ G' is 3-colorable