# Computational Models - Introduction[1]

**Handout Mode**


Instructors:

Prof. Yishay Mansour *mansour at cs.tau.ac.il*

Dr. Iftach Haitner *iftach.haitner at cs.tau.ac.il*

Teaching Assistants:

Mr. Mariano Schain *marianos at cs.tau.ac.il*

Mr. Yuval Moskovitch *mosyuval at gmail.com*


Tel Aviv University. Spring Semester, 2013. Mondays, 13–16 and Wednesdays, 10–13.

February 17/February 19, 2014

[1] Based on slides by Benny Chor, Tel Aviv University, modifying slides by Maurice Herlihy, Brown University.

Part I

# Administrativia

# Administrativia

Course website: http://tau-cm2014.wikidot.com/
Site (containing a forum) is our sole mean of disseminating information. Course Requirements:

- 6 problem sets (10% of grade).
  - Readable, concise, correct answers expected.
  - Late submission will not be accepted. (You have between one and two weeks, start working when you get them. Any excuse has to cover all the period.)
  - Solving problems independently is highly recommended.
  - A question from HW in final exam!
- Midterm, covering first 5 lectures (up to computability). We will discuss it when we get closer.
- Midterm scheduled to Friday, March 28, 2014 (final date).
- Final exam, covering all course material.

# AdministraTrivia II

- You must pass the final exam to get a passing course grade.
- Final Grade: 0.75*Exam + 0.15*Midterm + 0.10*HW.
- Prerequisites (formally): Extended introduction to computer science
  - But most importantly is “mathematical maturity”.
  - Students from other disciplines with mathematical background encouraged to contact the instructor.
- Textbook:
  - Michael Sipser, *Introduction to the theory of computation*, first or second editions.

Part II

# Course overview

# Why study theory?

- Basic computer science issues
  - What is a computation?
  - Are computers omnipotent?
  - What are the fundamental capabilities and limitations of computers?
- Pragmatic reasons
  - Avoid intractable or impossible problems.
  - Apply efficient algorithms when possible.
  - Learn to tell the difference.

# Course Topics

- Automata Theory: Basic model of computation.
  - Re-invented in many other disciplines.
- Computability Theory: What can computers do?
  - True impossibility results.
- Complexity Theory: What makes some problems computationally hard and others easy?
- Coping with intractability.

# Automata theory – Simple models

- Finite automata.
  - Related to controllers and hardware design.
  - Useful in text processing and finding patterns in strings.
  - Probabilistic (Markov) versions useful in modeling various natural phenomena (*e.g.* speech recognition).
- Push down automata.
  - Tightly related to a family of languages known as context free languages.
  - Play important role in compilers, design of programming languages, and studies of natural languages.

# Computability Theory

In the first half of the 20th century, mathematicians such as Kurt Göedel, Alan Turing, and Alonzo Church discovered that some fundamental problems cannot be solved by computers.

- Proof verification of statements can be automated.
- It is natural to expect that determining validity can also be done by a computer.
- Theorem: A computer cannot determine if mathematical statement true or false.
- Results needed theoretical models for computers.
- These theoretical models helped lead to the construction of real computers.

# Halting Problem

Can we decide if a program would halt on a given input?
Answer: No!

- Proof idea: Show a paradox assuming such a program exists
- Assume there exists a program DHALT(prog,input) that returns TRUE iff prog halts when run on input.
- Use DHALT to build program HALT that on input $x$:
  - Executes an infinite loop , if (program) $x$ halts on (input) $x$.
  - Halts, otherwise (i.e., if $x$ does not halt on $x$).
- What HALT(HALT) does?
- Hence, DHALT does not exist!

# Some Other Undecidable Problems

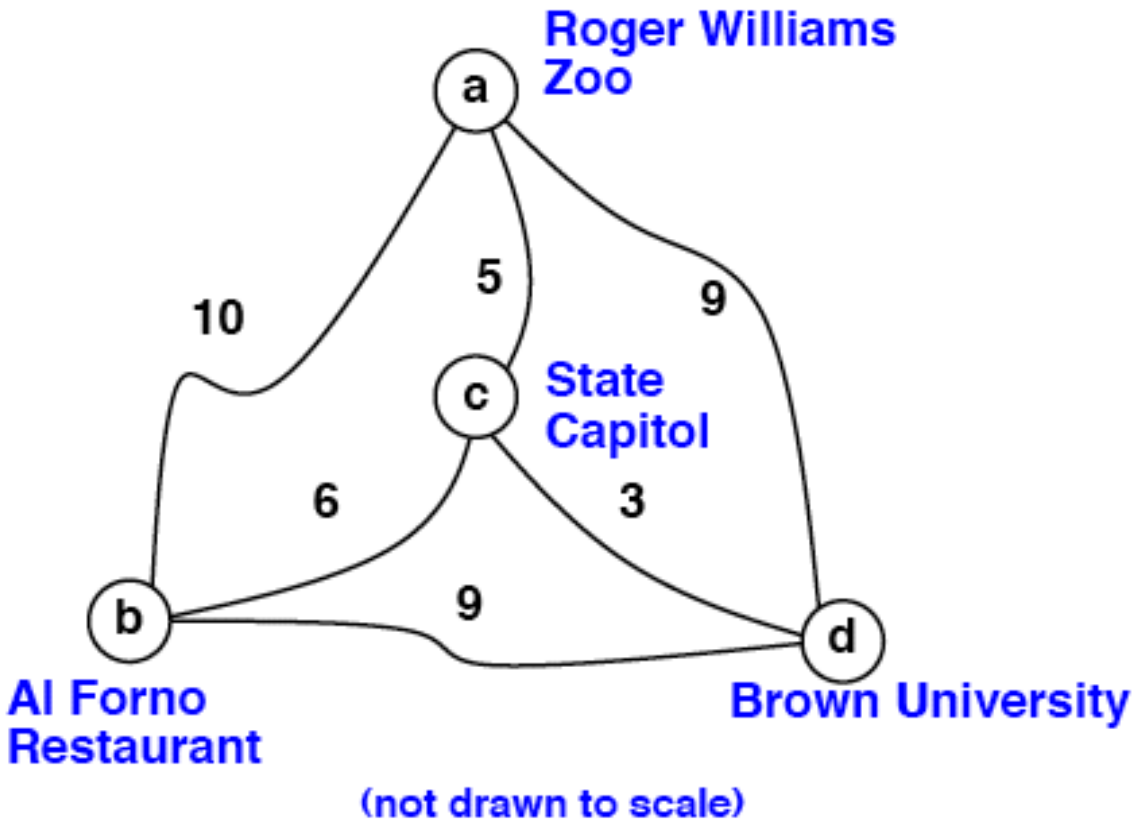
- Does a program run forever?
- Is a program correct?
- Are two programs equivalent?
- Is a program optimal (time wise – is it the fastest program solving a specific problem)?
- Does a polynomial equation with one or more variables and integer coefficients (e.g., $5x + 15y + 19xyz^2 = 12$) have an integer solution? (Hilbert's 10'th problem)
- How compressible is a given string?

# Complexity Theory

Key notion: tractable vs. intractable problems.

- A problem is a general computational question:
  - description of parameters
  - description of solution
- An algorithm is a step-by-step procedure
  - a recipe
  - a computer program
  - a mathematical object
- We want the most efficient algorithms
  - fastest (usually)
  - most economical with memory (sometimes)
  - expressed as a function of problem size

# Example: Traveling Salesman Problem

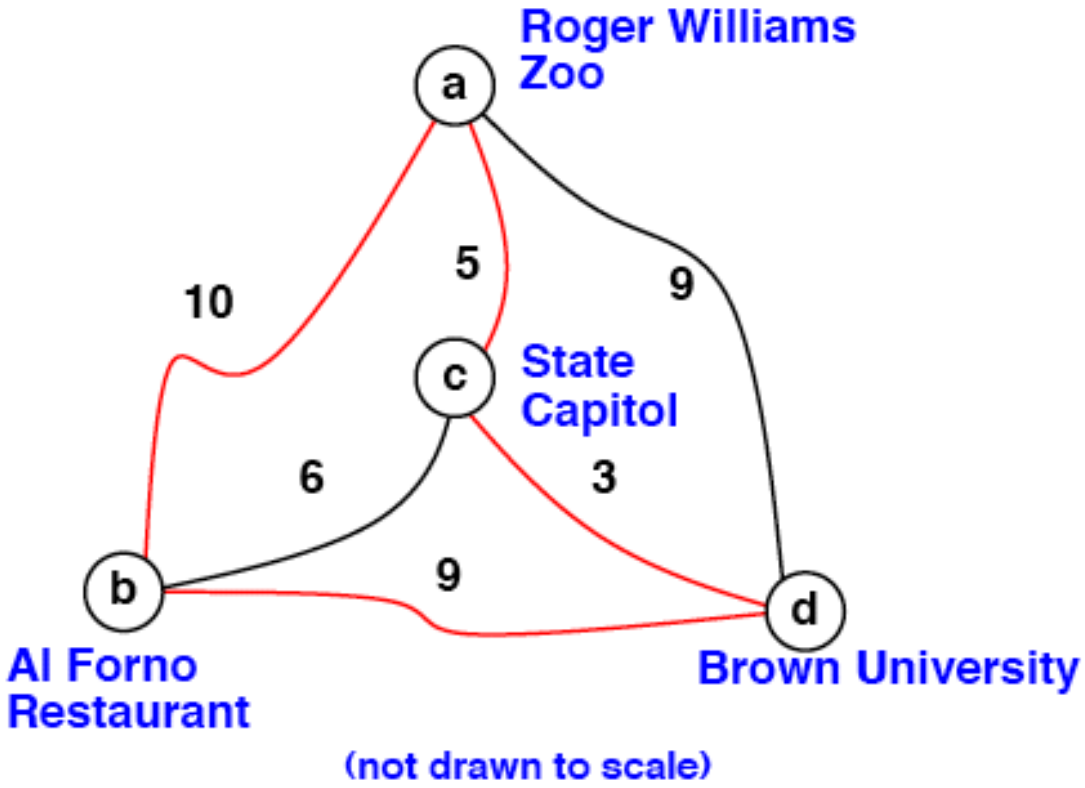



Input:

- set of cities
- set of inter-city distances

Goal: want the shortest tour through the cities

# Example: Traveling Salesman Problem




Example: $a, b, d, c, a$ has length 27

# Problem Size

- What is an appropriate measure of problem size?
  - $m$ nodes?
  - $m(m+1)/2$ distances?
- Encoding of a problem
  - alphabet of symbols
  - strings: $a/b/c/d//10/5/9//6/9//3$.
- Measures
  - Problem Size: length of encoding (above: 23 ascii characters).
  - Time Complexity: how long an algorithm runs, as function of problem size?

# Time Complexity - What is tractable?

- A function $f(n)$ is $O(g(n))$, if there is a constant $c$ such that for large enough $n$, $|f(n)| \leq c \cdot |g(n)|$.
- A polynomial-time algorithm is one whose time complexity is $O(p(n))$ for some polynomial $p(n)$, where $n$ denotes the length of the input.
- An exponential-time algorithm is one that runs in time $2^{cn}$ for some $c > 0$ (thus, not polynomial time ).

# Tractability – Basic distinction:

- Polynomial time $=$ tractable.
- Exponential time $=$ intractable.

| | 10 | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|---|
| $n$ | .00001 second | .00002 second | .00003 second | .00004 second | .00005 second | .00006 second |
| $n^2$ | .00001 second | .00004 second | .00009 second | .00016 second | .00025 second | .00036 second |
| $n^3$ | .00001 second | .00008 second | .027 second | .064 second | .125 second | .216 second |
| $n^5$ | .1 second | 3.2 seconds | 24.3 seconds | 1.7 minute | 5.2 minutes | 13.0 minutes |
| $2^n$ | .001 second | 1.0 second | 17.9 minutes | 12.7 days | 35.7 years | 366 centuries |
| $3^n$ | .059 second | 58 minutes | 6.5 years | 3855 centuries | $2 \cdot 10^8$ centuries | $1.3 \cdot 10^{13}$ centuries |

# Effect of Speed-Ups

Let's wait for faster hardware! Consider maximum problem size you can solve in an hour.

| | present | 100 times faster | 1000 times faster |
|---|---|---|---|
| $n$ | $N_1$ | $100N_1$ | $1000N_1$ |
| $n^2$ | $N_2$ | $10N_2$ | $31.6N_2$ |
| $n^3$ | $N_3$ | $4.64N_3$ | $10N_3$ |
| $n^5$ | $N_4$ | $2.5N_4$ | $3.98N_4$ |
| $2^n$ | $N_5$ | $N_5 + 6.64$ | $N_5 + 9.97$ |
| $3^n$ | $N_6$ | $N_6 + 4.19$ | $N_6 + 6.29$ |

# NP-Completeness / NP-Hardness

- Set of interesting and important optimization problems
- Exhaustive checking is exponential.
- All known algorithms are exponential
- Actual complexity not proven
- Believed to require exponential time

# Coping with NP-completeness

Intractability isn't everything.

- Find an approximate solution (is a solution within 10% of optimum good enough, ma'am?).
- Use randomization.
- Fixed parameter algorithms may be applicable.
- Heuristics can also help.

- Approximation, randomization, etc. are among the hottest areas in complexity theory and algorithmic research today.

Part III

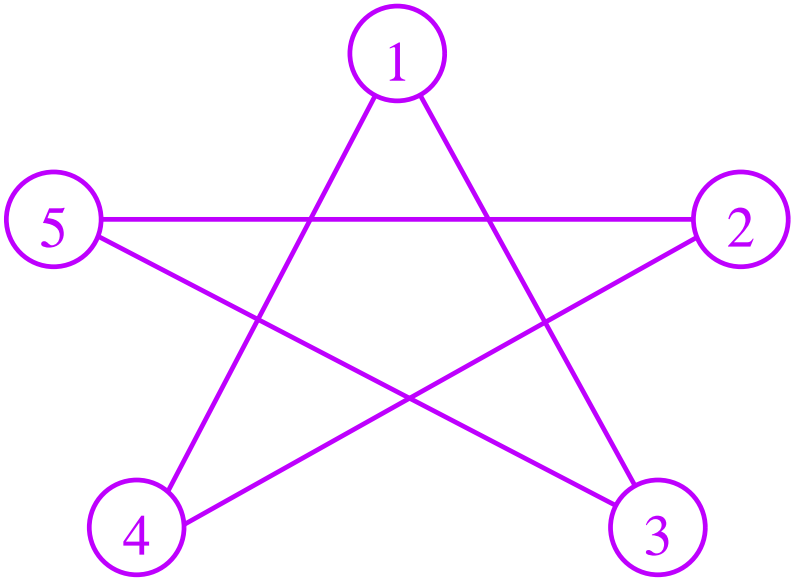
# A (Very) Short Math Review

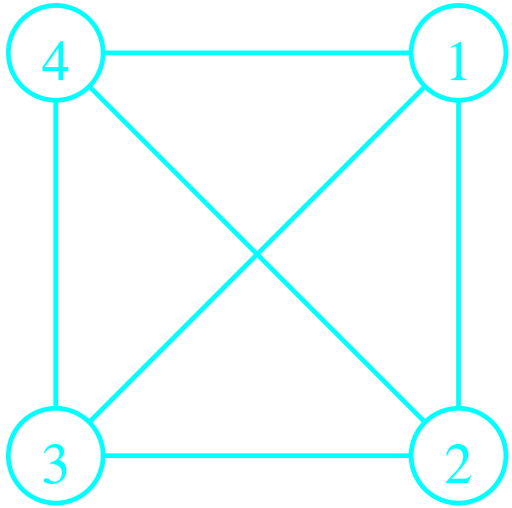
# A Very Short Math Review

- Graphs
- Strings and languages
- Mathematical proofs
- Mathematical notations (sets, sequences, ...) $\surd$
- Functions and predicates $\surd$

$\surd$ = will be done in recitation.

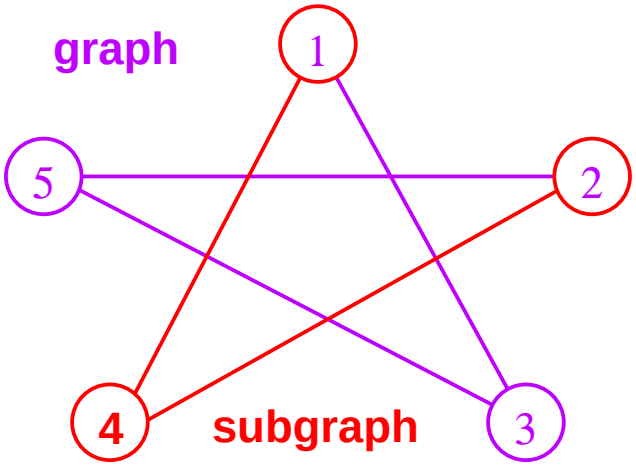
# Graphs

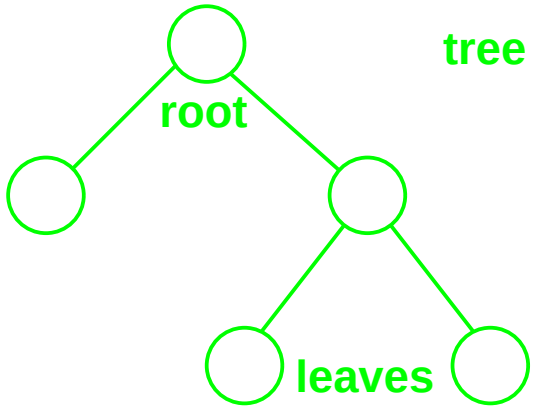


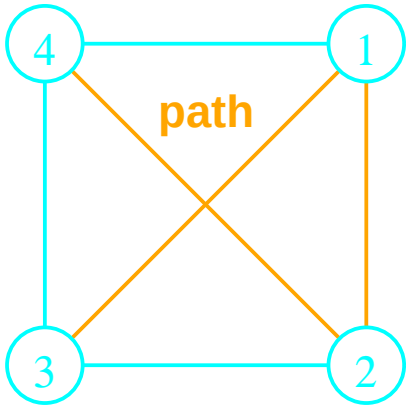



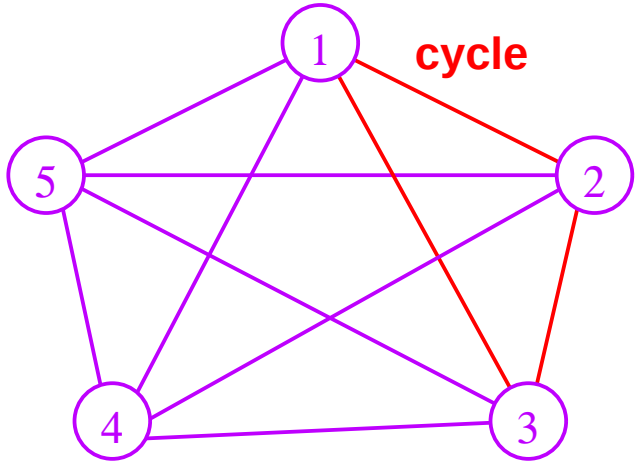

- $G = (V, E)$, where
  - $V$ is set of nodes or vertices, and
  - $E$ is set of edges
  - degree of a vertex is number of edges

# Graphs

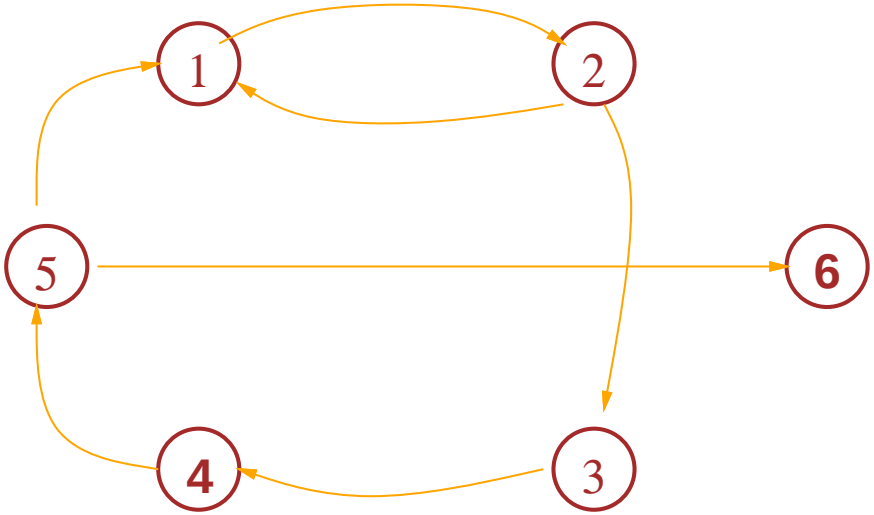

graph
1
5
2
4
subgraph
3

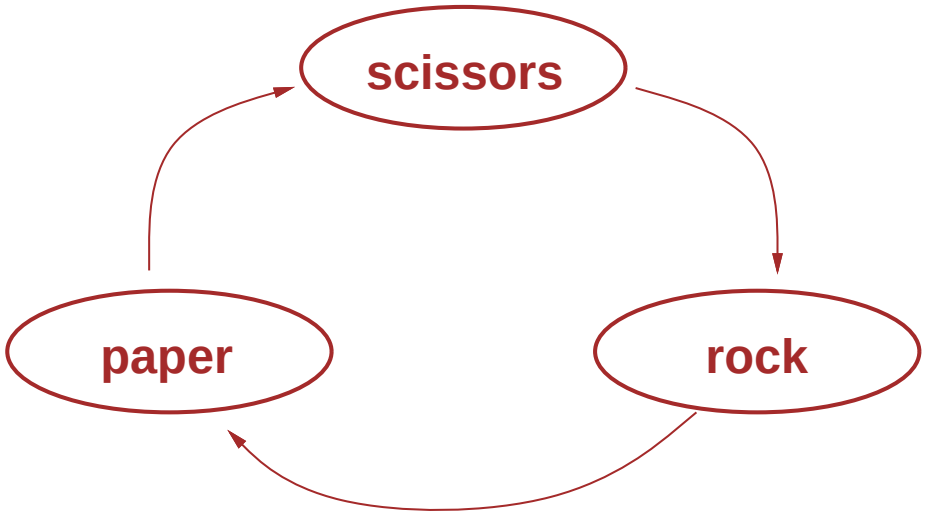

tree
root
leaves


4
1
path
3
2


1
cycle
5
2
4
3

# Directed Graphs



# Directed Graph and its Adjacency Matrix

Which directed graph is represented by the following 6-by-6 matrix?

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$A(i,j) = 1 \longleftrightarrow (i,j) \in E$

# Strings and Languages

- an alphabet is a finite set of symbols
- a string over an alphabet is a finite sequence of symbols from that alphabet.
- the length of a string is the number of symbols
- the empty string $\varepsilon$
- reverse: $abcd$ reversed is $dcba$.
- substring: $xyz$ in $xyzzy$.
- concatenation of $xyz$ and $zy$ is $xyzzy$.
- $x^k$ is $x \cdots x$, $k$ times.
- a language $\mathcal{L}$ is a (possibly infinite) set of strings.

# Proofs

We will use the following basic kinds of proofs.

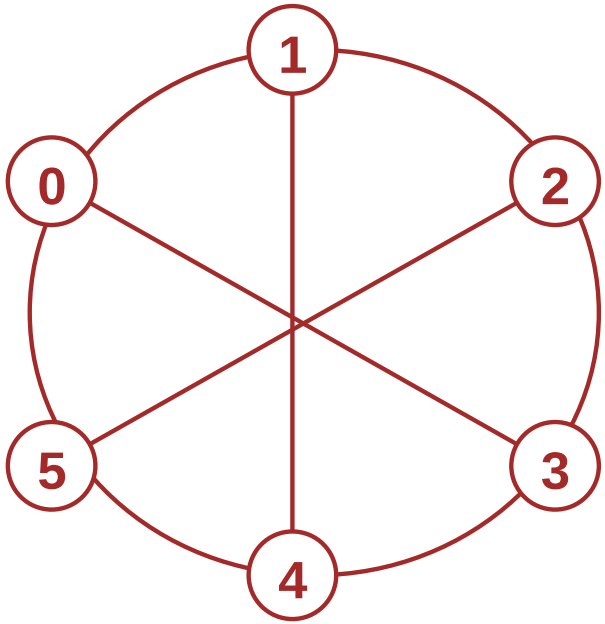
- by construction
- by contradiction
- by induction
- by reduction
- we will often mix them.

# Proof by Construction

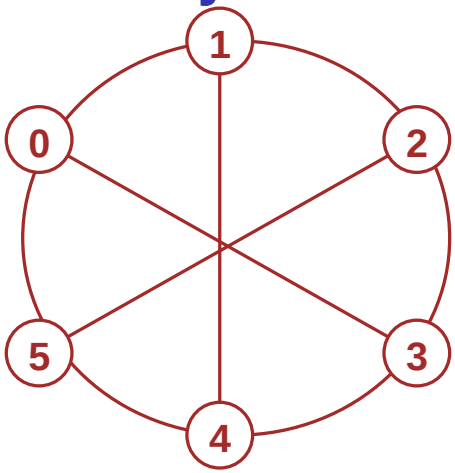
A graph is $k$-regular if every node has degree $k$.

**Theorem 1**

*For every even $n > 2$, there exists a 3-regular graph with $n$ nodes.*



# Proof by Construction




Proof: Construct $G = (V, E)$, where $V = \{0, 1, \ldots, n-1\}$ and

$$\begin{aligned} E &= \{\{i, i+1\}: \text{ for } 0 \le i \le n-2\} \cup \{n-1, 0\} \\ &\quad \cup \{\{i, i+n/2\}: \text{ for } 0 \le i \le n/2-1\}. \\ \mathit{degree}(i) &= |\{(i, i+1), (i, i-1), (i, i+n/2)\}| = 3 \end{aligned}$$

Missing details?

Note: A picture is helpful, but it is not a proof!

# Proof by Contradiction

**Theorem 2**

*$\sqrt{2}$ is irrational.*

Proof: Suppose not, then $\sqrt{2} = \frac{m}{n}$, where $m$ and $n$ are relatively prime.

$$n\sqrt{2} = m \Longrightarrow 2n^2 = m^2$$

So $m^2$ is even $\Longrightarrow$ $m$ is even (?). Let $m = 2k$.

$$2n^2 = (2k)^2 = 4k^2$$

$$\Longrightarrow n^2 = 2k^2$$

Thus $n^2$ is even, and so is $n$. Therefore both $m$ and $n$ are even, and not relatively prime!

# Proof by Induction

Prove properties of elements of an infinite set.

$$\mathbb{N} = \{1, 2, 3, \ldots\}$$

To prove that $\wp$ holds for each element, show:

- *base step:* show that $\wp(1)$ is true.
- induction step: show that if $\wp(i)$ is true (the induction hypothesis), then so is $\wp(i+1)$.

# Induction Example

**Theorem 3**

*All cows are the same color.*

Base step: A single-cow set is definitely the same color.
Induction Step: Assume all sets of $i$ cows are the same color.
Divide the set $\{1, \ldots, i+1\}$ into $U = \{1, \ldots, i\}$, and
$V = \{2, \ldots, i+1\}$.

All cows in $U$ are the same color by the induction hypothesis.
All cows in $V$ are the same color by the induction hypothesis.
All cows in $U \cap V$ are the same color by the induction hypothesis.

Hence, all cows are the same color.

*Quod Erat Demonstrandum* (QED).

# Induction Example (cont.)

(cows' images courtesy of www.crawforddirect.com/ cows.htm)

# Proof by Reduction

We can sometime solve problem $A$ by reducing it to problem $B$, whose solution we already know.

Example: $k^{th}$ element using $MAX$ element

Input: $S = \{7, 12, 3, 15, 9, 5, 19\}$, $k = 2$

Reduction to MAX

- For $i = 1, \ldots, k-1$ Do:
- $M = MAX(S)$
- $S = S \setminus \{M\}$
- Next $i$
- Output $MAX(S)$