

Computational Models

Introduction to the Theory of Computing

Instructor: **Prof. Yishay Mansour** (mansour at cs dot tau dot ac dot il)

Teaching Assistants: **Mr. Mariano Schain** (marianos at tau dot ac dot il)

Tel-Aviv University

Spring Semester, 2011. Wednesdays, 10–13.

Course site: <http://tau-cm.wikidot.com/>

Site (containing a forum) is our sole mean of disseminating information.

AdministraTrivia

Course Requirements:

- 6 problem sets
- **Readable, concise, correct** answers expected.
- Late submission will **not be accepted**.
- Solving problems **independently** is **highly recommended**.
- **Midterm**, covering first 5 lectures (up to computability).
- Midterm consists of multiple choice problems.
- **Final exam**, covering **all course material**.

Administra Trivia II

- You must pass the final exam to get a passing course grade.
- 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*, 1st or 2nd edition.

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:**
 - Approximation.
 - Randomization.
 - Fixed parameter algorithms.
 - Heuristics.

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.

- **Proof idea** Show a paradox using such a program HALT
- Simple Paradox:
 - The statement below is False
 - The statement above is True
 - $\text{HALT}(\text{Program}, \text{input})$ halts if program does not halt on input, otherwise does not halt.
- Consider construction of the type: $\text{HALT}(\text{HALT}, \text{HALT})$

Some Other Undecidable Problems

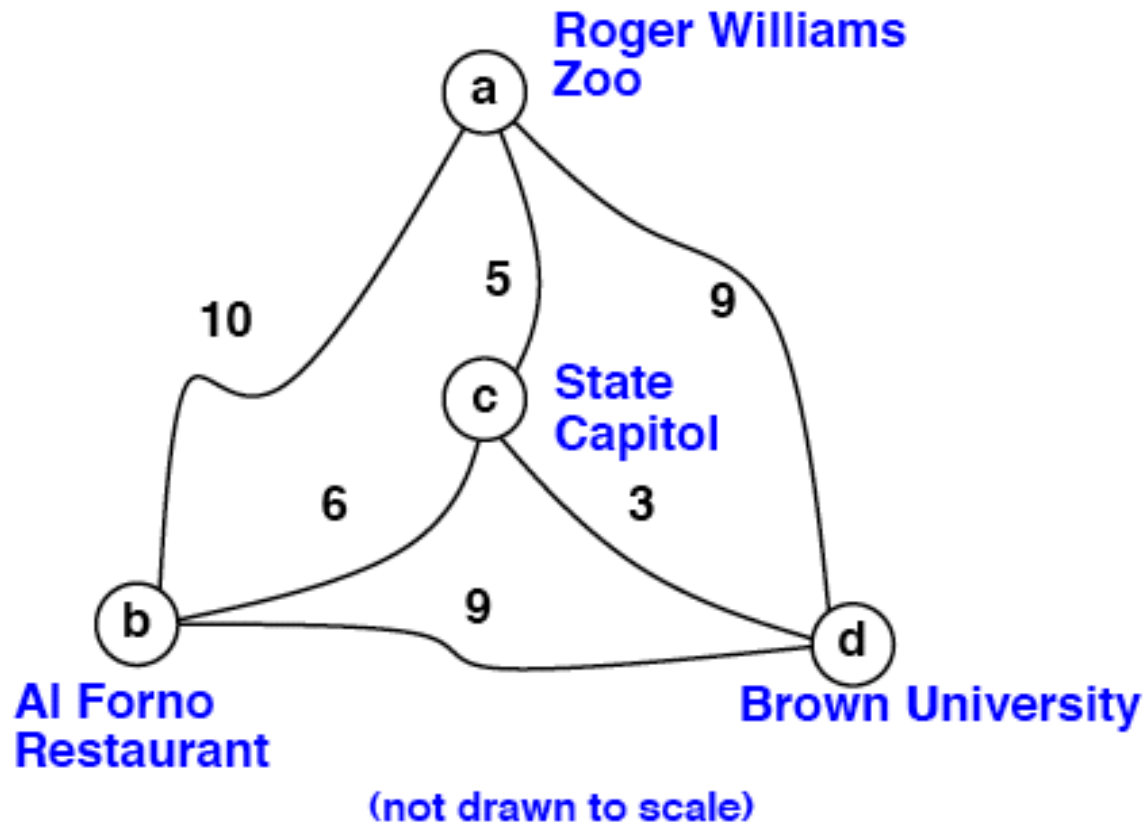
- Does a program run forever?
- Is a program correct?
- Are two programs equivalent?
- Is a program optimal (time wise – is it the fastest program solving a specific problem)?
- Does a polynomial equation with one or more variables and integer coefficients (e.g. $5x + 15y + 19xyz^2 = 12$) have an integer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

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

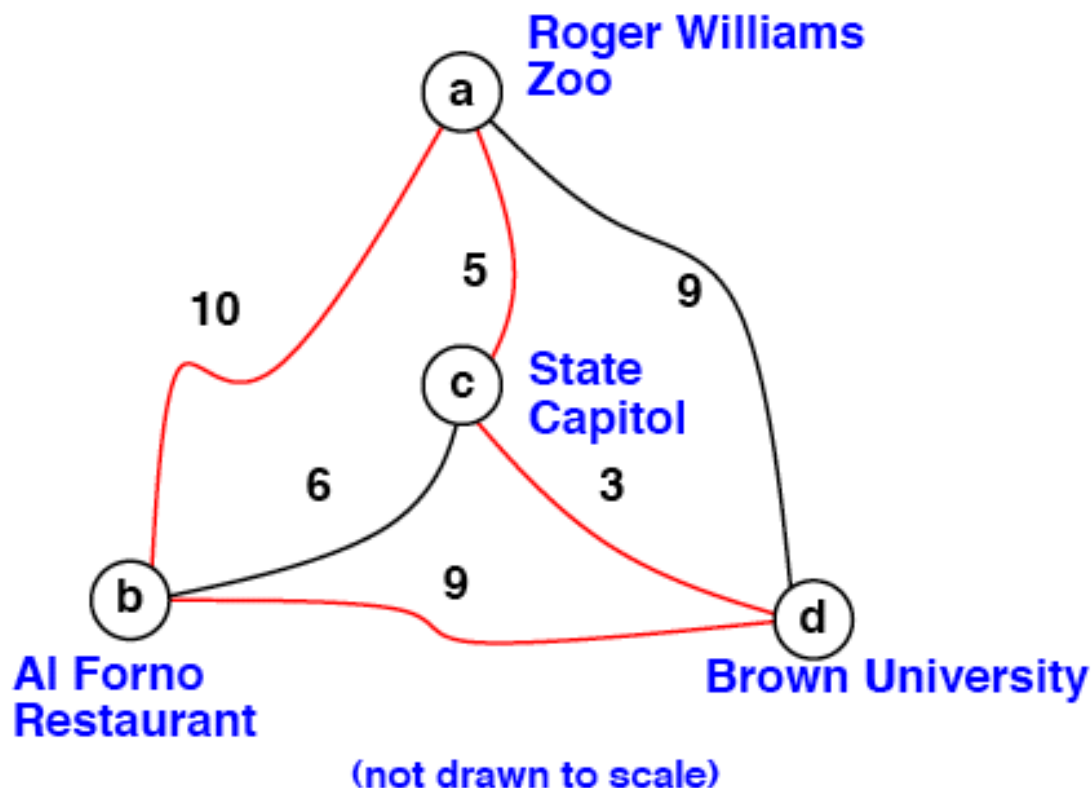
Example: Traveling Salesman Problem



Input:

- set of **cities**
- set of inter-city **distances**

Example: Traveling Salesman Problem



Goal:

- want the shortest tour through the cities
- 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?
- Use an **encoding** of the problem
 - alphabet of symbols
 - strings: a/b/c/d//10/5/9//6/9//3.
- Measures
 - **Problem Size**: length of encoding (here: 23 ascii characters).
 - **Time Complexity**: how long an algorithm runs, as function of problem size?

Time Complexity - What is tractable?

- We say that 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 whose time complexity cannot be bounded by a polynomial (e.g., $n^{\log n}$).

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

Your boss says:

“Get me an efficient traveling-salesman algorithm, or else...”

What are you going to do?

Response

“Yes Ma’am, expect it this afternoon!”

Problem is

- All known algorithms (essentially) check all possible paths.
- Exhaustive checking is exponential.
- Good luck!

Response

“Hah! I will prove that no such algorithm is possible”

Problem is, proving intractability is very hard.

Many important problems have

- no known tractable algorithms
- no known proof of intractability.

Response

“I can’t find an efficient algorithm.
I guess I’m just a pathetic loser. ”

- Bad for job security.

Response

“The problem is NP-hard. I can’t find an efficient algorithm, but neither can any of these famous people”

Advantage is:

- The problem is “just as hard” as other problems smart people can’t solve efficiently.
- So it would do your boss no good to fire you and hire a Technion/Hebrew Univ./MIT graduate.

Response

“Would you settle for a pretty good, but not the best, algorithm?”

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.

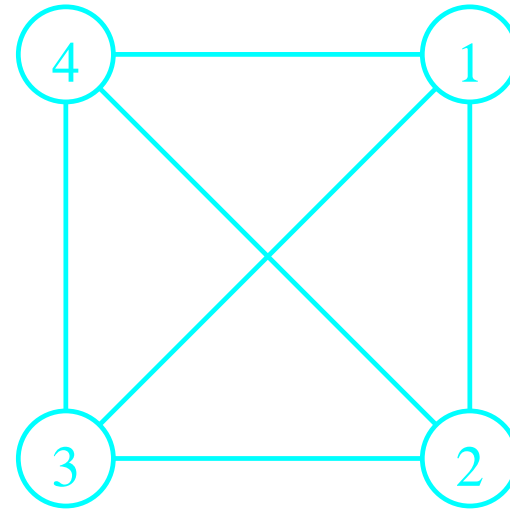
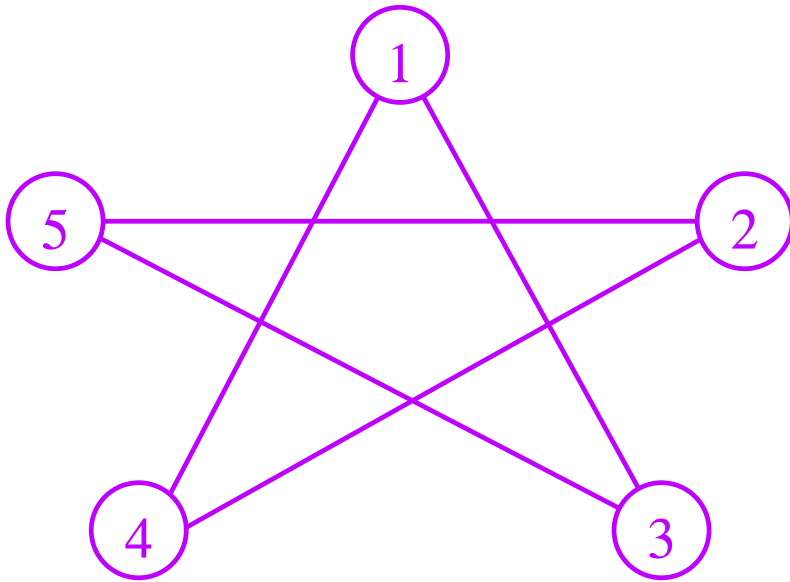
Next Subject

A Very Short Math Review

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

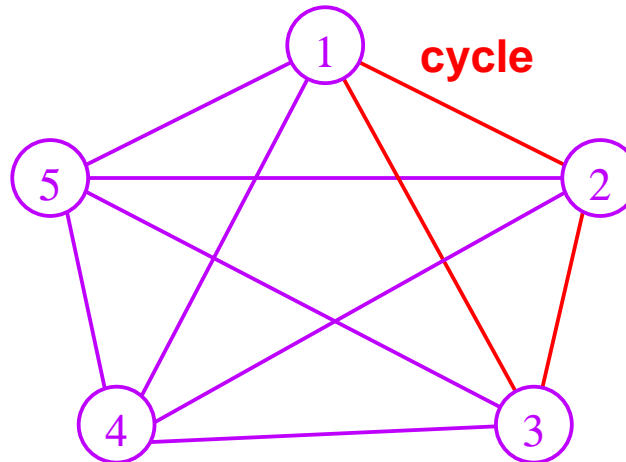
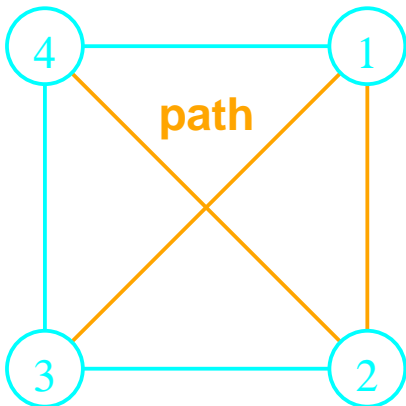
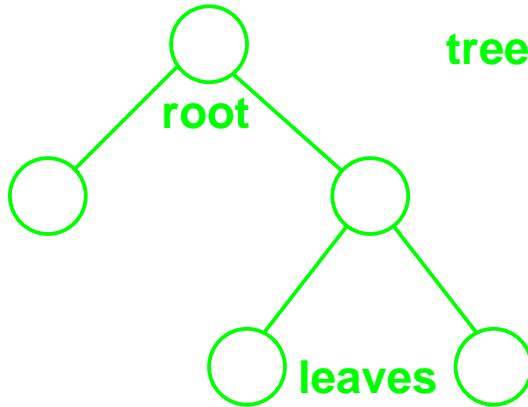
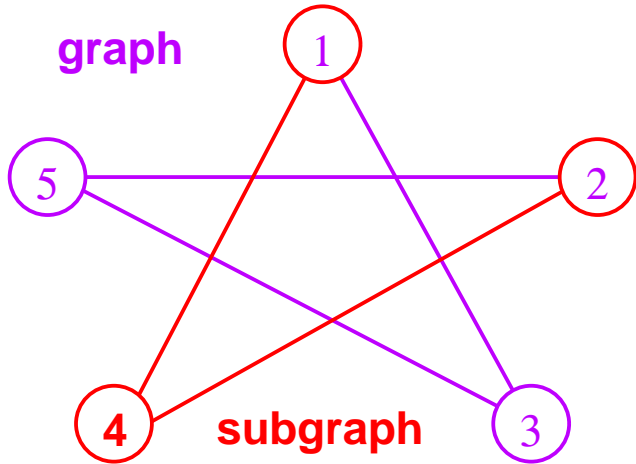
✓ = will be done in recitation.

Graphs

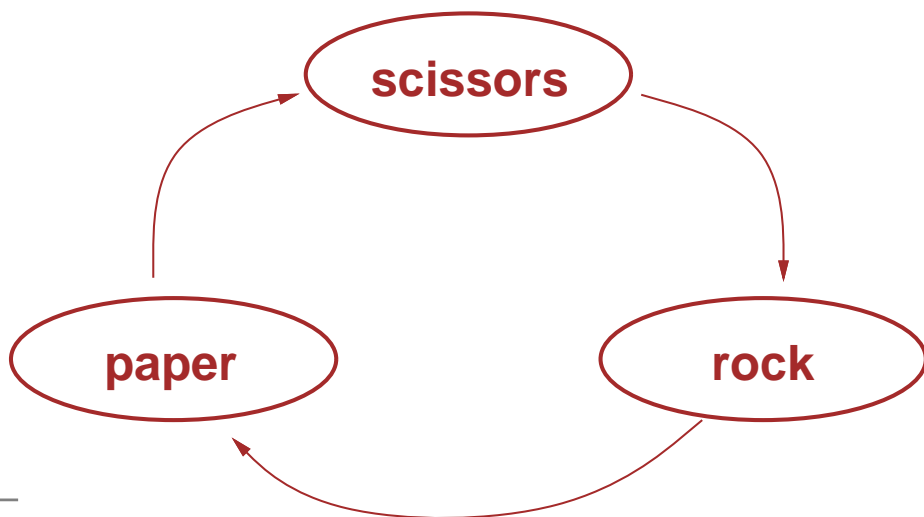
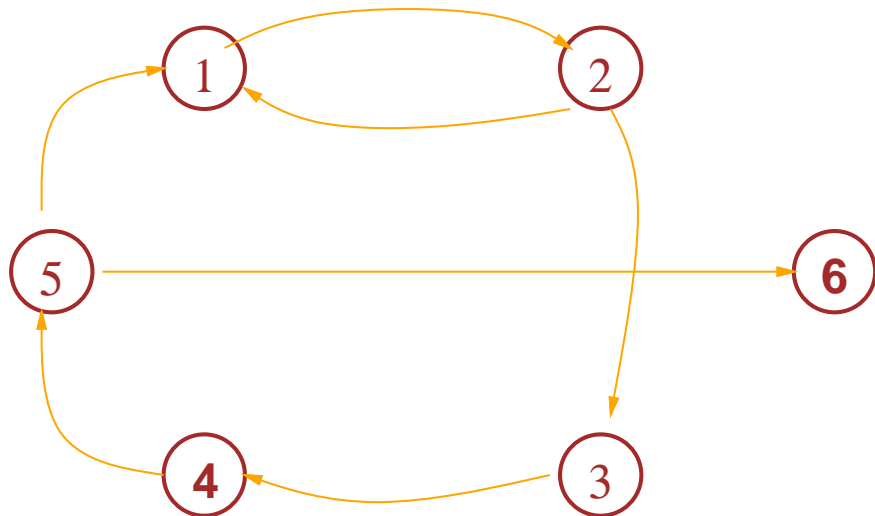


- $G = (V, E)$, where
- V is set of **nodes** or **vertices**, and
- E is set of **edges**
- **degree** of a vertex is number of edges

Graphs



Directed Graphs



Directed Graph and its Adjacency Matrix

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

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

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** ε
- **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** L is a set of strings.

Proofs

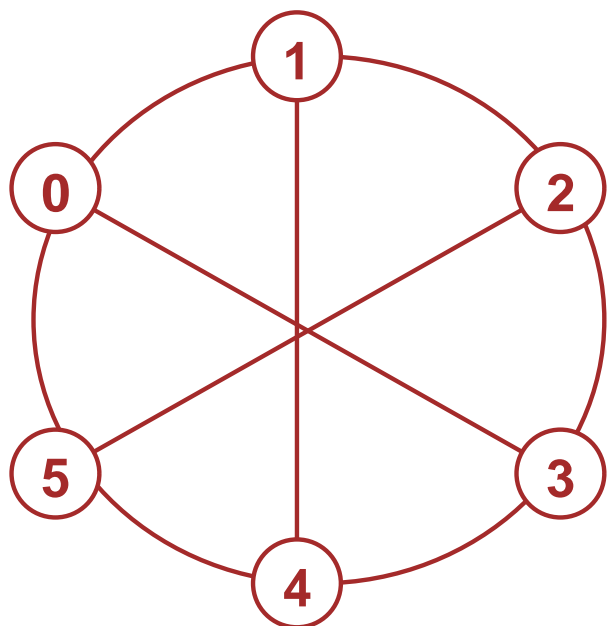
We will use the following basic kinds of proofs.

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

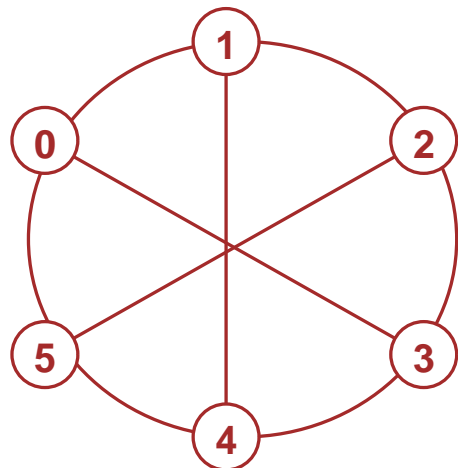
Proof by Construction

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

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



Proof by Construction



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

$$E = \{\{i, i + 1\} \mid \text{for } 0 \leq i \leq n - 2\} \cup \{n - 1, 0\} \\ \cup \{\{i, i + n/2\} \mid \text{for } 0 \leq i \leq n/2 - 1\}.$$

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

Proof by Contradiction

Theorem: $\sqrt{2}$ is irrational.

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

$$\begin{aligned}n\sqrt{2} &= m \\2n^2 &= m^2\end{aligned}$$

Proof by Contradiction (cont.)

So m^2 is even, and so is $m = 2k$.

$$\begin{aligned}2n^2 &= (2k)^2 \\ &= 4k^2 \\ n^2 &= 2k^2\end{aligned}$$

Thus n^2 is even, and so is n .

Therefore both m and n are even, and not relatively prime!
Reductio ad absurdam.

Proof by Induction

Prove properties of elements of an infinite set.

$$\mathcal{N} = \{1, 2, 3, \dots\}$$

To prove that φ holds for each element, show:

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

Induction Example

Theorem: 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 $\{1, \dots, i + 1\}$ into $U = \{1, \dots, i\}$, and $V = \{2, \dots, i + 1\}$.

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.

Induction Example (cont.)

Ergo, all cows are the same color.

Quod Erat Demonstrandum (QED).

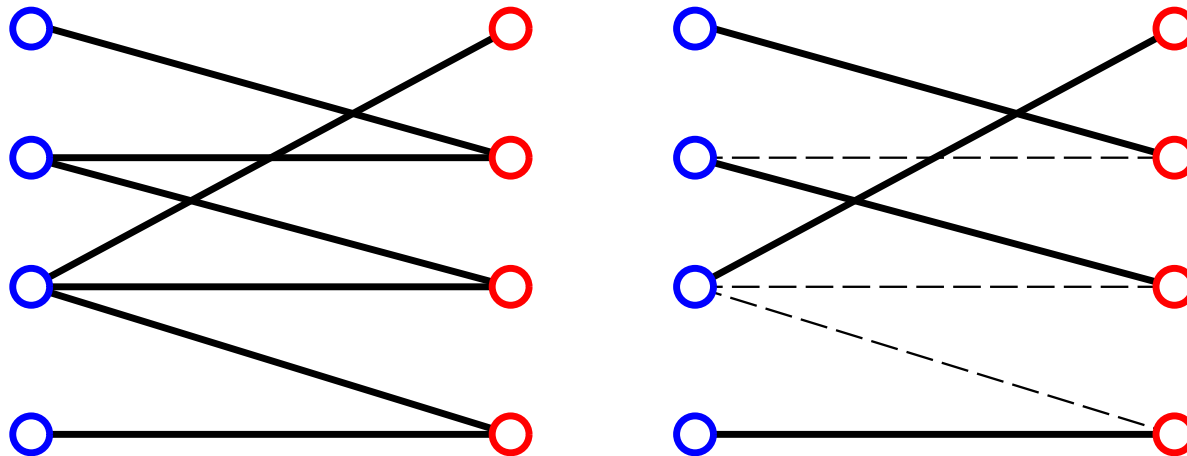


(cows' images courtesy of www.crawforddirect.com/cows.htm)

Proof by Reduction

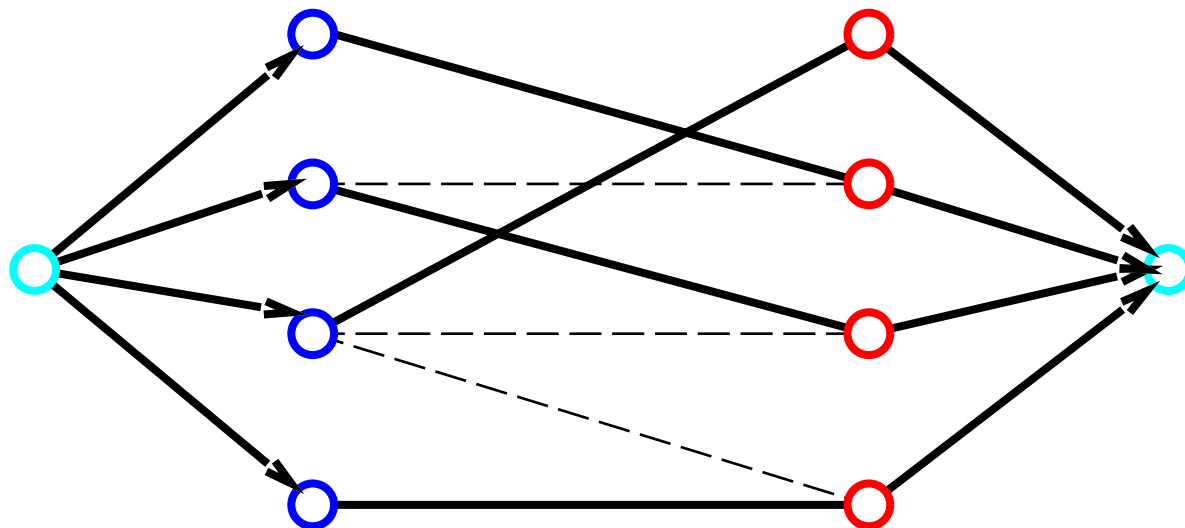
We can sometime solve problem **A** by **reducing** it to problem **B**, whose solution we already know.

Example: Maximal matching in bipartite graphs:



Proof by Reduction

Reducing bipartite matching to MAX FLOW:



Reduction: Put **capacity 1** on each edge.

Maximum flow corresponds to **maximum matching**. So if we have an algorithm that produces max flow, we can easily derive a maximum bipartite matching from it.