A Math Review

• Mathematical notations
• Mathematical proofs
• Functions and predicates
• Graphs
Sets

- order doesn’t matter
Sets

- order doesn’t matter
- finite: \{2, 4, 6\}
Sets

- order doesn’t matter
- finite: \( \{2, 4, 6\} \)
- infinite: \( \{2, 4, 6, \ldots \} \ \{x \mid x \text{ is even}\} \)
Sets

- order doesn’t matter
- finite: \( \{2, 4, 6\} \)
- infinite: \( \{2, 4, 6, \ldots \} \) \( \{x \mid x \text{ is even}\} \)
- subset: \( U \subset V, \ U \subseteq V \)
Sets

- order doesn’t matter
- finite: \{2, 4, 6\}
- infinite: \{2, 4, 6, \ldots\} \{x \mid x \text{ is even}\}
- subset: \(U \subset V\), \(U \subseteq V\)
- union: \(U \cup V\)
Sets

- order doesn’t matter
- finite: \{2, 4, 6\}
- infinite: \{2, 4, 6, \ldots\} \{x \mid x \text{ is even}\}
- subset: \(U \subset V, U \subseteq V\)
- union: \(U \cup V\)
- intersection: \(U \cap V\)
Sets

- order doesn’t matter
- finite: \{2, 4, 6\}
- infinite: \{2, 4, 6, \ldots\} \{x \mid x \text{ is even}\}
- subset: \(U \subset V, U \subseteq V\)
- union: \(U \cup V\)
- intersection: \(U \cap V\)
- power set of \(U\): \(2^U\), is set of all subsets

Slides modified by Benny Chor, based on original slides by Maurice Herlihy, Brown University.
Sets

- order doesn’t matter
- finite: \{2, 4, 6\}
- infinite: \{2, 4, 6, \ldots\} \{x \mid x \text{ is even}\}
- subset: \(U \subset V, U \subseteq V\)
- union: \(U \cup V\)
- intersection: \(U \cap V\)
- power set of \(U\): \(2^U\), is set of all subsets
- cardinalities: \(\aleph_0, \aleph_1, 2^{\aleph_0}\)
Sets

- order doesn’t matter
- finite: \{2, 4, 6\}
- infinite: \{2, 4, 6, \ldots\} \{x \mid x \text{ is even}\}
- subset: \( U \subset V, U \subseteq V \)
- union: \( U \cup V \)
- intersection: \( U \cap V \)
- power set of \( U \): \( 2^U \), is set of all subsets
- cardinalities: \( \aleph_0, \aleph_1, 2^{\aleph_0} \)
- the continuum hypothesis
Sets

- order doesn’t matter
- finite: \{2, 4, 6\}
- infinite: \{2, 4, 6, \ldots\} \{x \mid x \text{ is even}\}
- subset: \(U \subset V, U \subseteq V\)
- union: \(U \cup V\)
- intersection: \(U \cap V\)
- power set of \(U\): \(2^U\), is set of all subsets
- cardinalities: \(\aleph_0, \aleph_1, 2^{\aleph_0}\)
- the continuum hypothesis
- etc., etc.
Sequences

- order matters
Sequences

- order matters
- \((1, 3, 5, \ldots)\)
Sequences

- order matters
- \((1, 3, 5, \ldots)\)
- finite sequence called a *tuple*
Sequences

- order matters
- $(1, 3, 5, \ldots)$
- finite sequence called a *tuple*
- 2 elements is a *pair*
Sequences

- order matters
- \((1, 3, 5, \ldots)\)
- finite sequence called a *tuple*
- 2 elements is a *pair*
- \(k\) elements is a *\(k\)-tuple*
Functions and Predicates

A function or mapping

\[ f : X \rightarrow Y \]

\[ f : \text{domain} \rightarrow \text{range} \]
Functions and Predicates

A function or mapping

\[ f : X \rightarrow Y \]

- \( f : \text{domain} \rightarrow \text{range} \)
- input values are called \textit{arguments}.
Functions and Predicates

A function or mapping

\[ f : X \rightarrow Y \]

- \( f : domain \rightarrow range \)
- input values are called arguments.
- a \( k \)-ary function has \( k \) arguments.
Functions and Predicates

A function or mapping

\[ f : X \rightarrow Y \]

- \( f : \text{domain} \rightarrow \text{range} \)
- Input values are called \textit{arguments}.
- A \( k \)-ary function has \( k \) arguments.
- A \textit{predicate} is a function with range \{true, false\}.
Functions and Predicates

A function or mapping

\[ f : X \rightarrow Y \]

- \( f : \text{domain} \rightarrow \text{range} \)
- input values are called arguments.
- a \( k \)-ary function has \( k \) arguments.
- a predicate is a function with range \{true, false\}.
- a binary relation is a predicate whose domain is a Cartesian product \( U \times V \).
A Popular Example

The “SCISSORS, PAPER, ROCK” game as a predicate (from the "row player" point of view):

<table>
<thead>
<tr>
<th></th>
<th>SCISSORS</th>
<th>PAPER</th>
<th>ROCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCISSORS</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>PAPER</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>ROCK</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

and as a relation:

\{ (\text{SCISSORS, PAPER}), (\text{PAPER, ROCK}), (\text{ROCK, SCISSORS}) \}.
Graphs

\[ G = (V, E), \text{ where} \]
Graphs

\[ G = (V, E), \text{ where} \]

- \( V \) is set of nodes or vertices, and
Graphs

\[ G = (V, E), \text{ where} \]
\[ V \text{ is set of nodes or vertices, and} \]
\[ E \text{ is set of edges} \]
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
- Subgraph
- Tree
- Root
- Leaves
- Path
- Cycle

Slides modified by Benny Chor, based on original slides by Maurice Herlihy, Brown University.
Directed Graphs

```
1 -> 2
5 <- 6
4 -> 3
```

```
scissors

paper -> rock
```

Slides modified by Benny Chor, based on original slides by Maurice Herlihy, Brown University.
A Directed Graph and its Adjacency 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*
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.
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
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* $\varepsilon$
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 $dcba$. 
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 \( xyzzy \).
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 $xyzzy$. 
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 $xyzzz$.
- Concatenation of $xyz$ and $zy$ is $xyzzy$.
- $x^k$ is $x \cdots x$, $k$ times.
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 $L$ is a set of strings.
Proofs

We will use the following basic kinds of proofs.

- by construction
Proofs

We will use the following basic kinds of proofs.

- by construction
- by contradiction
Proofs

We will use the following basic kinds of proofs.

- by construction
- by contradiction
- by induction
Proofs

We will use the following basic kinds of proofs.

- by construction
- by contradiction
- by induction
- by reduction
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

$V = \{0, 1, \ldots, n - 1\}$ 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.

\[
\begin{align*}
  n\sqrt{2} &= m \\
  2n^2 &= m^2
\end{align*}
\]
Proof by Contradiction (cont.)

So \( m^2 \) is even, and so is \( m = 2k \).

\[
2n^2 = (2k)^2 \\
= 4k^2 \\
n^2 = 2k^2
\]

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.

\[ \mathcal{N} = \{1, 2, 3, \ldots\} \]

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

- **base step**: show that \( \wp(1) \) is true.
Proof by Induction

Prove properties of elements of an infinite set. 
\[ \mathcal{N} = \{1, 2, 3, \ldots\} \]

To prove that \( \varphi \) 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.
Induction Example

**Theorem:** All cows are the same color.

**Base step:** A single-cow set is definitely the same color.
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 $\{1, \ldots, i+1\}$ into $U = \{1, \ldots, i\}$, and $V = \{2, \ldots, i+1\}$. 

Slides modified by Benny Chor, based on original slides by Maurice Herlihy, Brown University.
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 \(\{1, \ldots, i + 1\}\) into \(U = \{1, \ldots, i\}\), and \(V = \{2, \ldots, i + 1\}\). All cows in \(U\) are the same color by the induction hypothesis.
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 $\{1, \ldots, i + 1\}$ 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.
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 $\{1, \ldots, i + 1\}$ 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.
Induction Example (cont.)

*Ergo*, all cows are the same color.

*Quod Erat Demonstrandum.*
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.