Graphs

Problem Set One due right now in the box up front.
Mathematical Structures

• Just as there are common data structures in programming, there are common mathematical structures in discrete math.

• So far, we've seen simple structures like sets and natural numbers, but there are many other important structures out there.

• Over the next few weeks, we'll explore several of them.
Graphs
Chemical Bonds
PANFLUTE FLOWCHART

1. Do you need one?
   - Yes: No you don't.
   - No: No panflute.
Me too!
What's in Common

• Each of these structures consists of
  • Individual objects and
  • Links between those objects.

• Goal: find a general framework for describing these objects and their properties.
A **graph** is a mathematical structure for representing relationships.
A **graph** is a mathematical structure for representing relationships.

A graph consists of a set of **nodes** (or **vertices**) connected by **edges** (or **arcs**).
A **graph** is a mathematical structure for representing relationships.

A graph consists of a set of **nodes** (or **vertices**) connected by **edges** (or **arcs**).
A graph is a mathematical structure for representing relationships.

A graph consists of a set of nodes (or vertices) connected by edges (or arcs).
Some graphs are **directed**.
Some graphs are **undirected**.
Some graphs are **undirected**.

You can think of them as directed graphs with edges both ways.
Formalizing Graphs

- How might we define a graph mathematically?
- Need to specify
  - What the nodes in the graph are, and
  - What the edges are in the graph.
- The nodes can be pretty much anything.
- What about the edges?
Ordered and Unordered Pairs

- An **unordered pair** is a set \( \{a, b\} \) of two elements (remember that sets are unordered).
  - \( \{0, 1\} = \{1, 0\} \)
- An **ordered pair** \((a, b)\) is a pair of elements in a specific order.
  - \((0, 1) \neq (1, 0)\).
  - Two ordered pairs are equal iff each of their components are equal.
Formalizing Graphs

- Formally, a **graph** is an ordered pair $G = (V, E)$, where
  - $V$ is a set of nodes.
  - $E$ is a set of edges.
- $G$ is defined as an *ordered* pair so it's clear which set is the nodes and which is the edges.
- $V$ can be any set whatsoever.
- $E$ is one of two types of sets:
  - A set of *unordered* pairs of elements from $V$.
  - A set of *ordered* pairs of elements from $V$. 
Undirected Connectivity
Navigating a Graph

- IP
- PC
- VC
- PT
- CI
- CC
- SC
- VEC
- FC
- LT
Navigating a Graph

From

To
Navigating a Graph
Navigating a Graph

From PT to VC to PC to CC to SC to CDC
Navigating a Graph

From T to PT

From CI to VEC

To CC

From VC to CDC

From CDC to SC

From SC to LT

From VEC to FC
Navigating a Graph

PT → VC → VEC → SC → CDC
Navigating a Graph

PT → CI → FC → CDC
A path from \( v_1 \) to \( v_n \) is a sequence of nodes \( v_1, v_2, \ldots, v_n \) where \( (v_k, v_{k+1}) \in E \) for all natural numbers in the range \( 1 \leq k \leq n - 1 \).

The length of a path is the number of edges it contains, which is one less than the number of nodes in the path.
A path from \( v_1 \) to \( v_n \) is a sequence of nodes \( v_1, v_2, \ldots, v_n \) where \( \{v_k, v_{k+1}\} \in E \) for all natural numbers in the range \( 1 \leq k \leq n - 1 \).

The length of a path is the number of edges it contains, which is one less than the number of nodes in the path.
Navigating a Graph
Navigating a Graph

Nodes:
- IP
- PC
- VC
- PT
- CI
- CC
- CDC
- SC
- VEC
- FC
- LT
Navigating a Graph

PC → CC → VEC → VC → PC
PC → CC → VEC → VC → PC → CC → VEC → VC → PC
Navigating a Graph

From

To
Navigating a Graph
Navigating a Graph

PT → VC → PC → CC → VEC → VC → IP
A cycle in a graph is a path from a node to itself.

The length of a cycle is the number of edges in that cycle.
A **simple path** in a graph is a path that does not revisit any nodes or edges.

A **simple cycle** in a graph is a cycle that does not revisit any nodes or edges (except the start/end node).
Navigating a Graph
Navigating a Graph
Navigating a Graph
Navigating a Graph
Navigating a Graph
In an undirected graph, two nodes $u$ and $v$ are called \textbf{connected} iff there is a path from $u$ to $v$.

We denote this as $u \leftrightarrow v$.

If $u$ is not connected to $v$, we write $u \leftrightarrow v$. 
Next Time

• The Rest of The Lecture
  • Sorry about the fire alarm!
  • Connected components.
  • Planar graphs.

• Binary Relations
  • Equivalence relations.
  • Partial orders (ITA).