Introduction to Graphs

Introduction

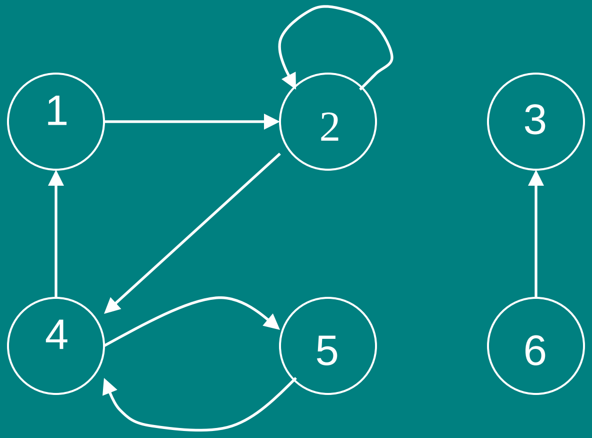
- Graphs are a generalization of trees
  - Nodes or verticies
  - Edges or arcs
- Two kinds of graphs
  - Directed
  - Undirected

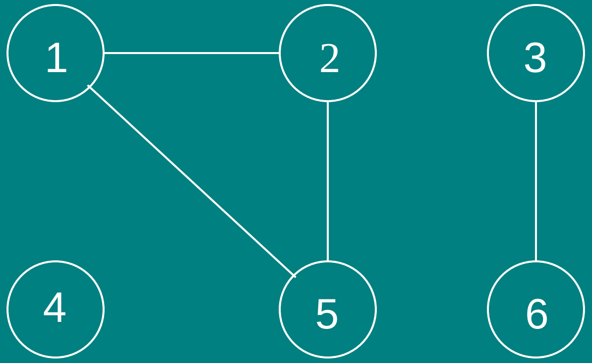
Introduction: Formal Definition

- A graph $G = (V,E)$ consists of a finite set of vertices, V, and a finite set of edges E.
- Each edge is a pair (v,w) where $v, w \in V$

What is a Graph?

- Informally a *graph* is a set of nodes joined by a set of lines or arrows.





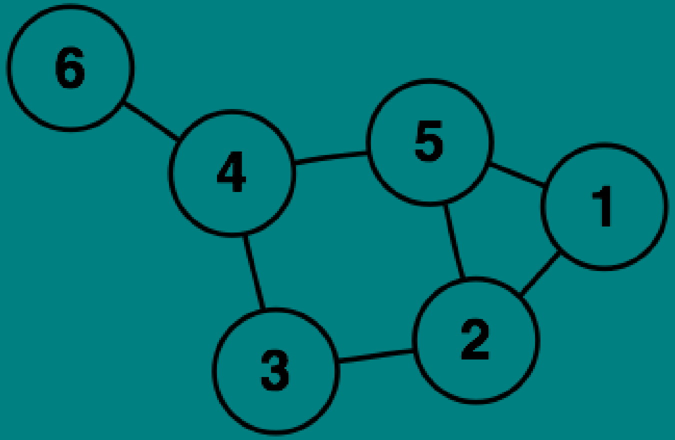
Definition: Graph

- G is an ordered triple G:=(V, E, f)
  - V is a set of nodes, points, or vertices.
  - E is a set, whose elements are known as edges or lines.
  - f is a function
    - maps each element of E
    - to an unordered pair of vertices in V.

Definitions

- Vertex
  - Basic Element
  - Drawn as a *node* or a *dot*.
  - V**ertex set** of *G* is usually denoted by *V*(*G*), or *V*
- Edge
  - A set of two elements
  - Drawn as a line connecting two vertices, called end vertices, or endpoints.
  - The edge set of G is usually denoted by E(G), or E.

Example



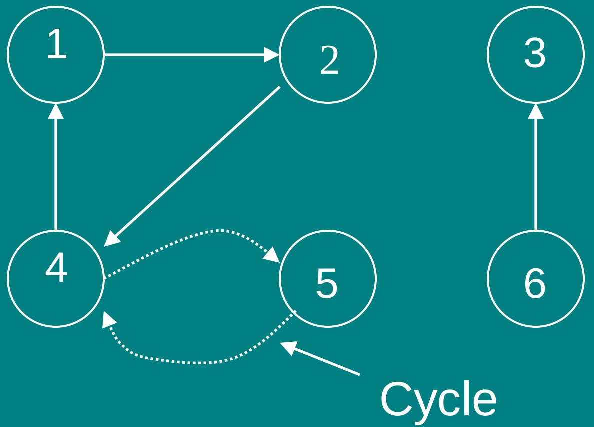
- V:={1,2,3,4,5,6}
- E:={{1,2},{1,5},{2,3},{2,5},{3,4},{4,5},{4,6}}

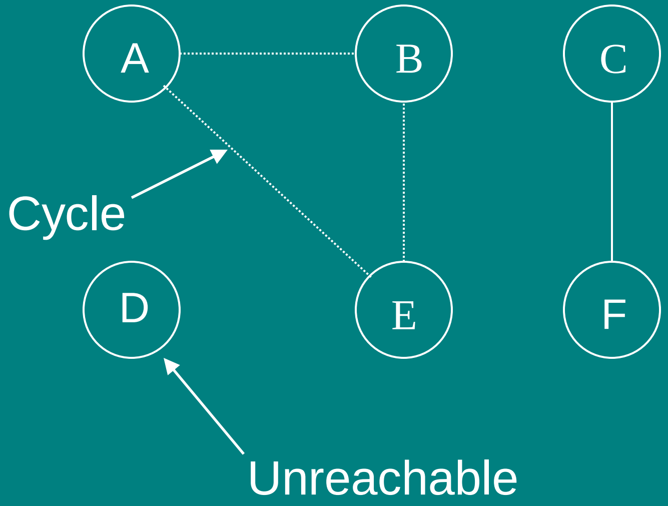
Simple Graphs

Simple graphs are graphs without multiple edges or self-loops.

Path

- A *path* is a sequence of vertices such that there is an edge from each vertex to its successor.
- A path is ***simple*** if each vertex is distinct.





Simple path from 1 to 5
= [1, 2, 4, 5]
Our text's alternates the vertices and edges.

If there is path *p* from *u* to *v* then we say *v* is **reachable** from *u* via *p*.

Introduction: Formal Definition

- A **directed** graph, or **digraph**, is a graph in which the edges are ordered pairs
  - $(v, w) \neq (w, v)$
- An **undirected** graph is a graph in which the edges are unordered pairs
  - $(v, w) == (w, v)$

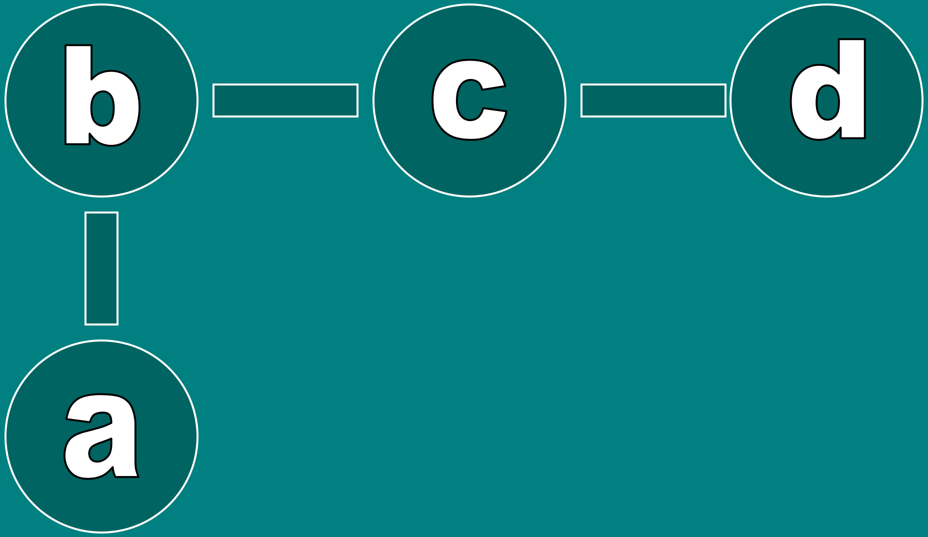
Introduction: Directed Graphs

- In a directed graph, the edges are arrows.
- Directed graphs show the flow from one node to another and not vise versa.

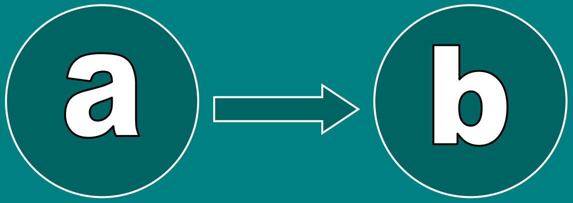
$$\begin{tikzcd}
B \arrow[r] & C & D \arrow[l] \\
A \arrow[u] & &
\end{tikzcd}$$

Introduction: Undirected Graphs

- In a undirected graph, the edges are lines.
- Undirected graphs show a relationship between two nodes.

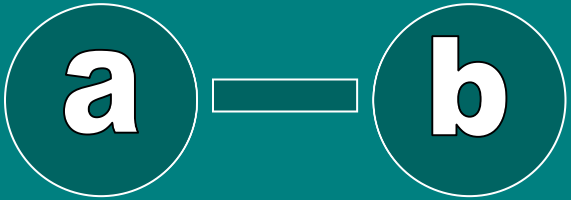


Terminology



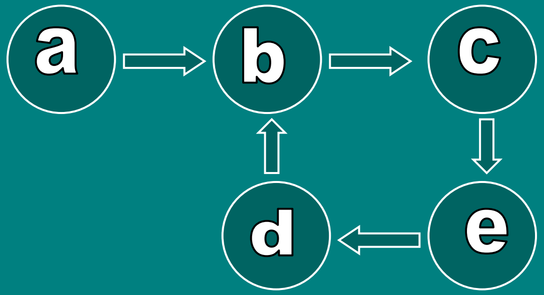
- In the directed graph above, b is **adjacent** to a because (a, b) $\in$ E. Note that a is *not* adjacent to b.
- A is a **predecessor** of node B
- B is a **successor** of node A
- The **source** of the edge is node A, the **target** is node B

Terminology



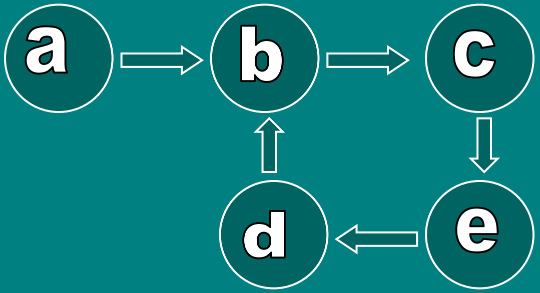
- In the undirected graph above, a and b are **adjacent** because (a,b) $\in$ E. a and b are called **neighbors.**

Terminology



- A **path** is a sequence of vertices w_1, w_2,…w_n such that $(w_i, w_{i+1}) \in E$, $1 <= i < n$, and each vertex is unique except that the path may start and end on the same vertex
- The **length** of the path is the number of edges along the path

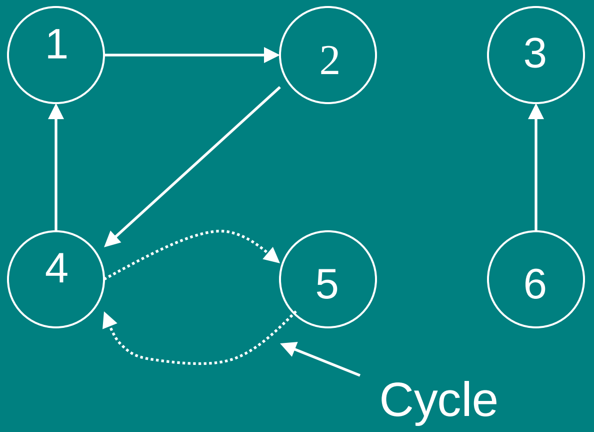
Terminology

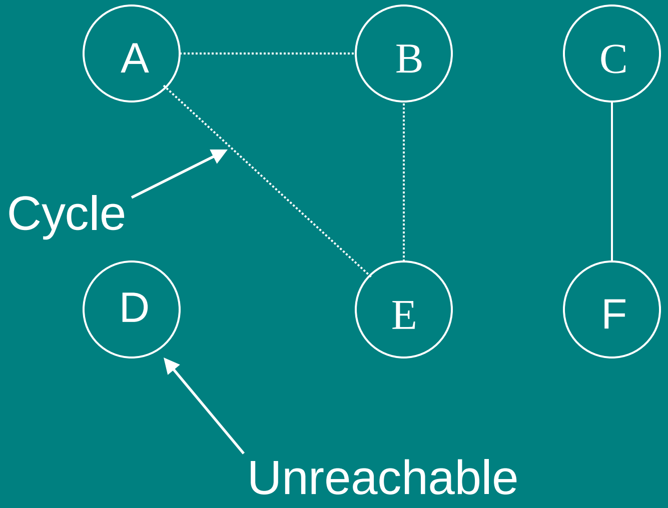

$$\begin{tikzcd}
a \arrow[r] & b \arrow[r] & c \arrow[d] \\
 & d \arrow[u] & e \arrow[l]
\end{tikzcd}$$

- An **acyclic path** is a path where each vertex is unique
- A **cyclic path** is a path such that
  - There are at least two vertices on the path
  - $w_1 = w_n$ (path starts and ends at same vertex)

Cycle

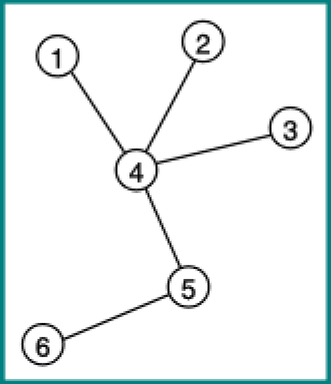
- A path from a vertex to itself is called a ***cycle***.
- A graph is called ***cyclic*** if it contains a cycle;
  - otherwise it is called ***acyclic***





Tree

- Connected Acyclic Graph
- Two nodes have *exactly* one path between them



Test Your Knowledge

Cyclic or Acyclic?

1.

$$\begin{tikzcd}
a \arrow[r] \arrow[d] & b \arrow[d] \arrow[dr] & \\
c \arrow[r] & d & e \arrow[l] \\
& f \arrow[u] &
\end{tikzcd}$$

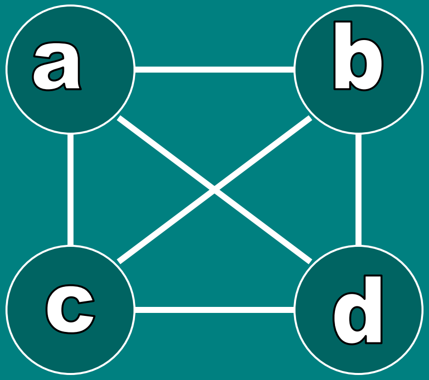
2.

$$\begin{tikzcd}
a \arrow[r] \arrow[d] & b \arrow[dr] & \\
c \arrow[r] & d \arrow[u] \arrow[d] & e \arrow[l] \arrow[dl] \\
& f &
\end{tikzcd}$$

Terminology

- A directed graph that has *no* cyclic paths is called a **DAG** (a Directed Acyclic Graph).
- An undirected graph that has an edge between every pair of vertices is called a **complete** graph.

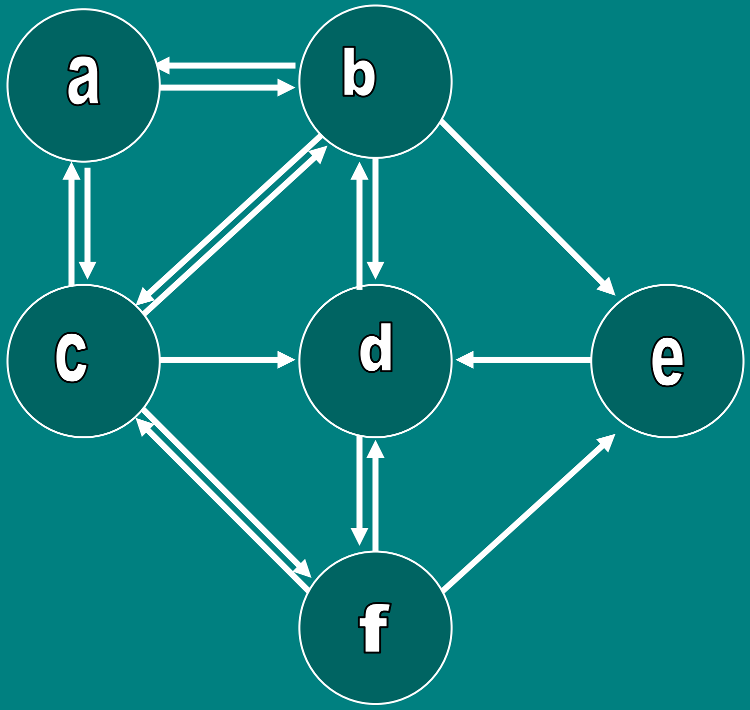
Note: A directed graph can also be a complete graph; in that case, there must be an edge from every vertex to every other vertex.



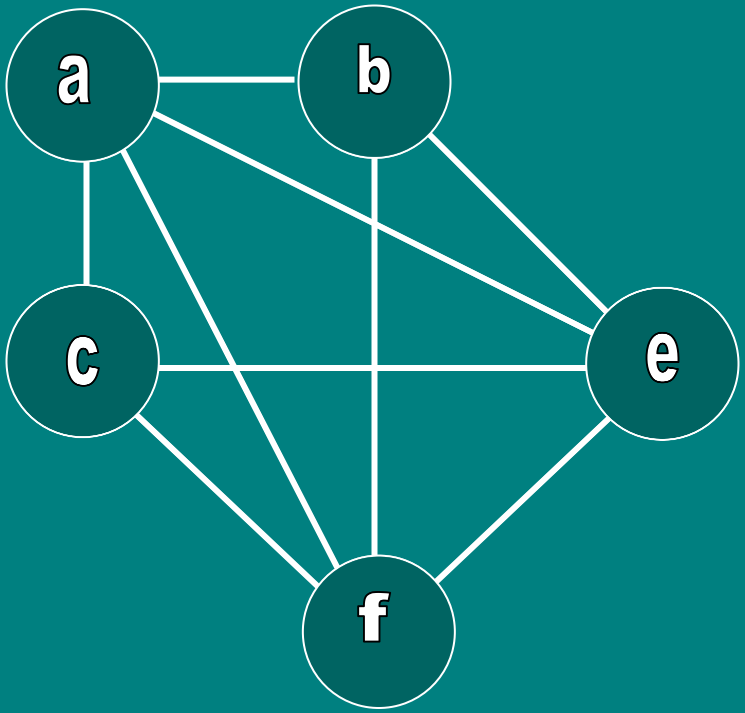
Test Your Knowledge

Complete, or "Acomplete" (Not Complete)

1.



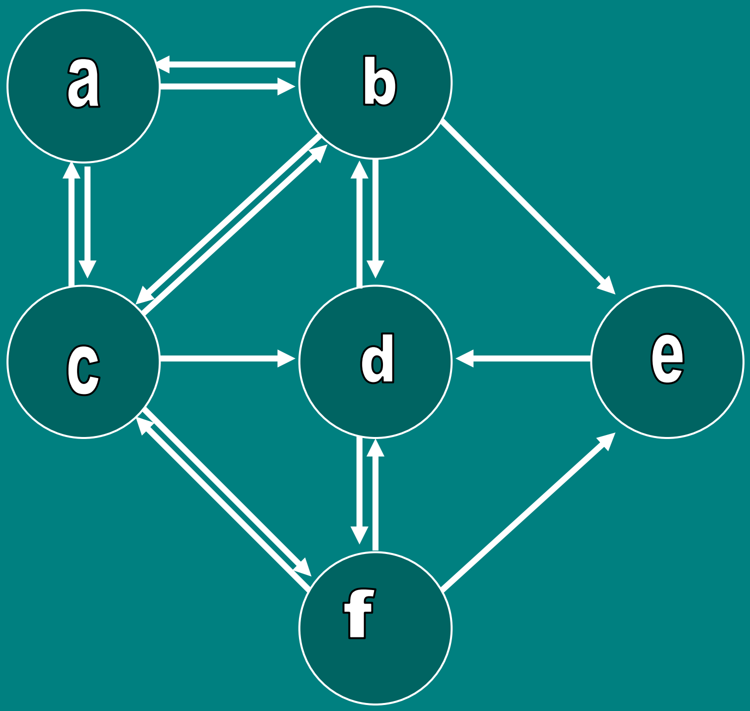
2.



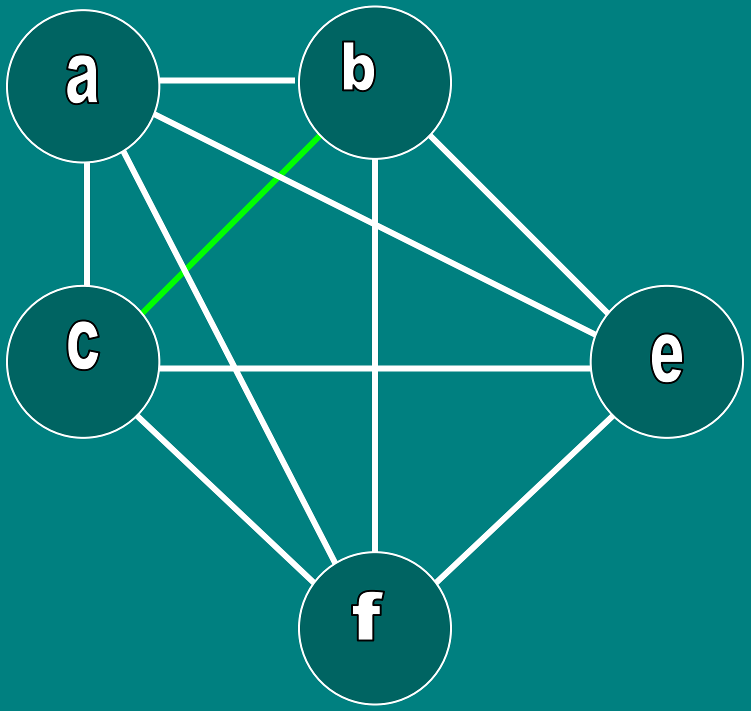
Test Your Knowledge

Complete, or “Acomplete” (Not Complete)

1.



2.



Terminology

- An undirected graph is **connected** if a path exists from every vertex to every other vertex
- A directed graph is **strongly connected** if a path exists from every vertex to every other vertex
- A directed graph is **weakly connected** if a path exists from every vertex to every other vertex, disregarding the direction of the edge

Test Your Knowledge

Connected, Strongly connected, or Weakly connected

1.

$$\begin{tikzcd}
a \arrow[r] & b \arrow[r] & c \arrow[d] \\
 & d \arrow[u] & e \arrow[l]
\end{tikzcd}$$

2.

$$\begin{tikzcd}
a \arrow[r] & b \arrow[d] & c \arrow[d] \\
 & d \arrow[ul] \arrow[ur] & e \arrow[l]
\end{tikzcd}$$

3.

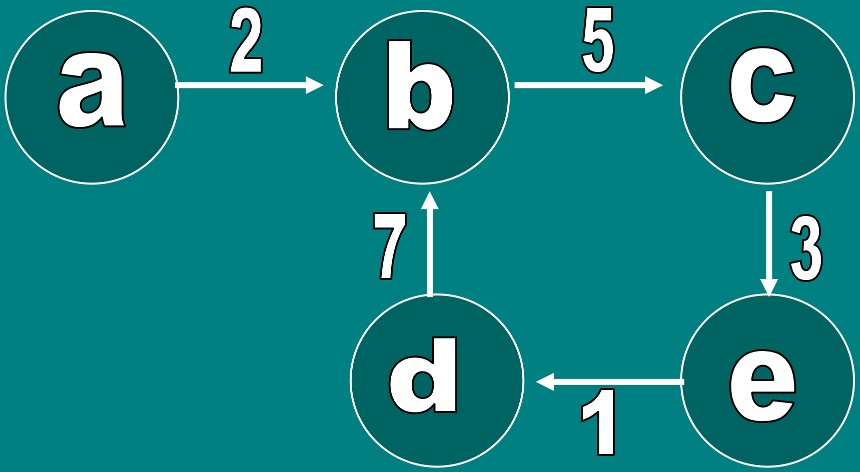
$$\begin{tikzcd}
a \arrow[dr, dash] & b \arrow[r, dash] \arrow[dr, dash] & c \arrow[d, dash] \\
 & d & e
\end{tikzcd}$$

4.

$$\begin{tikzcd}
a \arrow[r, dash] & b \arrow[r, dash] \arrow[d, dash] \arrow[dr, dash] & c \\
 & d & e
\end{tikzcd}$$

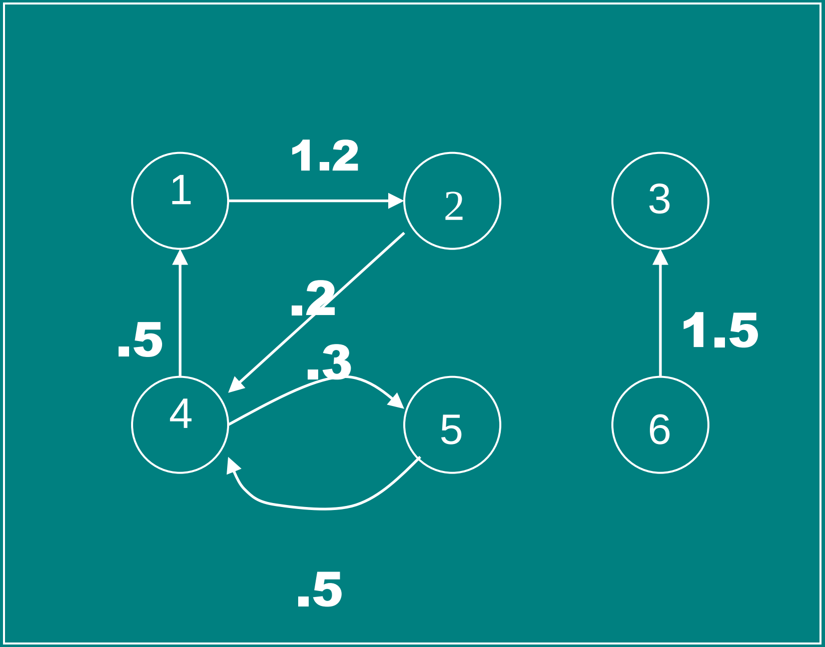
Terminology

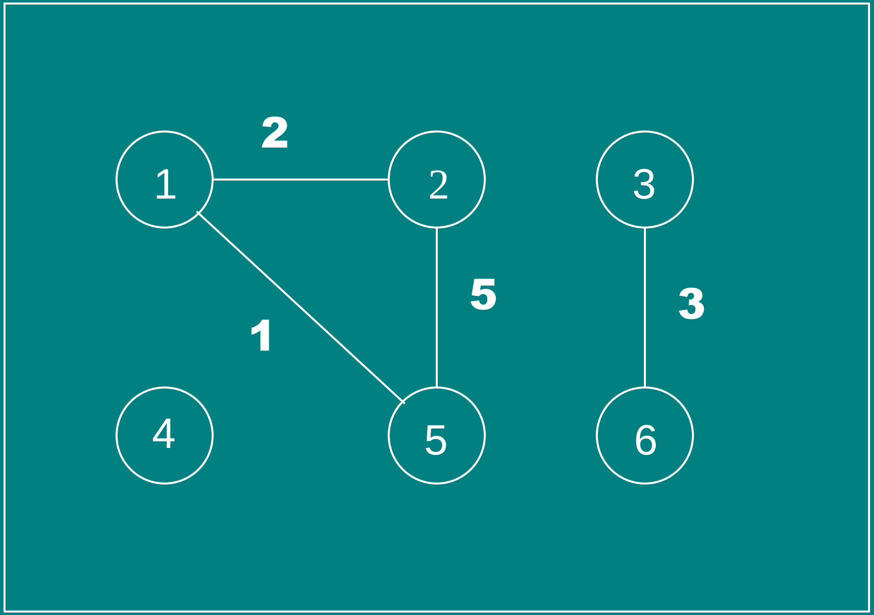
- A graph is known as a **weighted graph** if a weight or metric is associated with each edge.


$$\begin{tikzcd}
a \arrow[r, "2"] & b \arrow[r, "5"] & c \arrow[d, "3"] \\
 & d \arrow[u, "7"] & e \arrow[l, "1"]
\end{tikzcd}$$

A ***weighted graph***

- is a graph for which each edge has an associated ***weight***, usually given by a ***weight function*** $w: E \to \mathbf{R}$.





Uses for Graphs

- **Computer network:** The set of vertices V represents the set of computers in the network. There is an edge (u, v) if and only if there is a direct communication link between the computers corresponding to u and v.

Uses for Graphs

- **Precedence Constraints:** Suppose you have a set of jobs to complete, but some must be completed before others are begun. (For example, Atilla advises you always pillage before you burn.) Here the vertices are jobs to be done. Directed edges indicate constraints; there is a directed edge from job u to job v if job u must be done before job v is begun.

Uses for Graphs

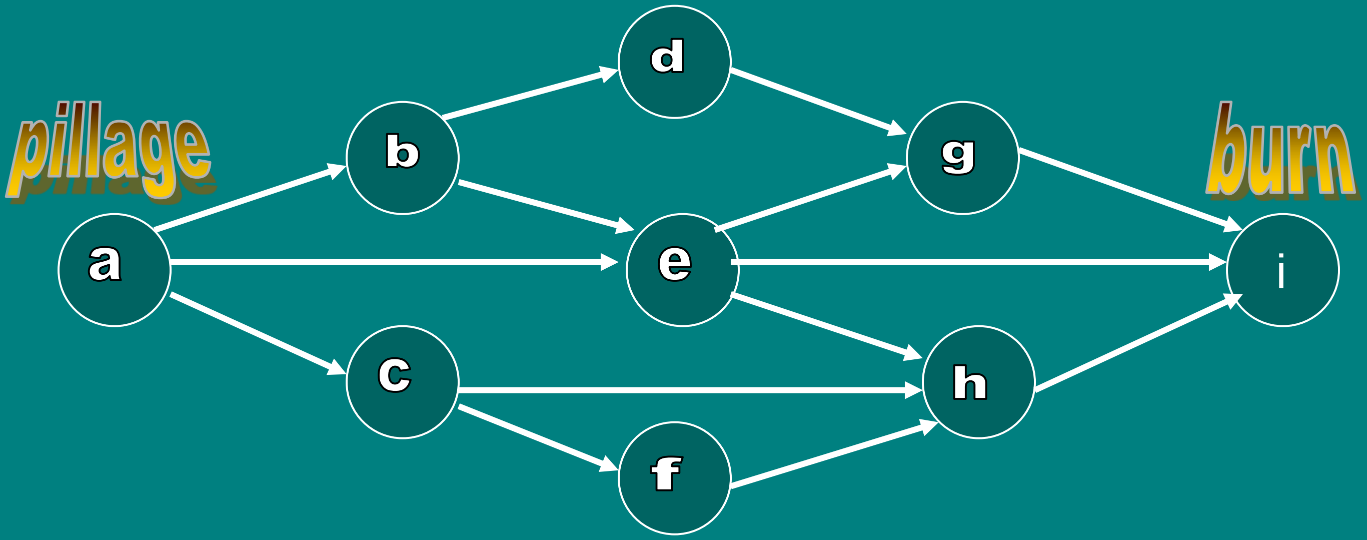
- **Two-Player Game Tree:** All of the possibilities in a board game like chess can be represented in a graph. Each vertex stands for one possible board position. (For chess, this is a very big graph!)

Topological Sort

Don’t burn before you pillage!

Topological Sort

- Informally, a topological sort is a linear ordering of the vertices of a DAG in which all successors of any given vertex appear in the sequence after that vertex.



Method to the Madness

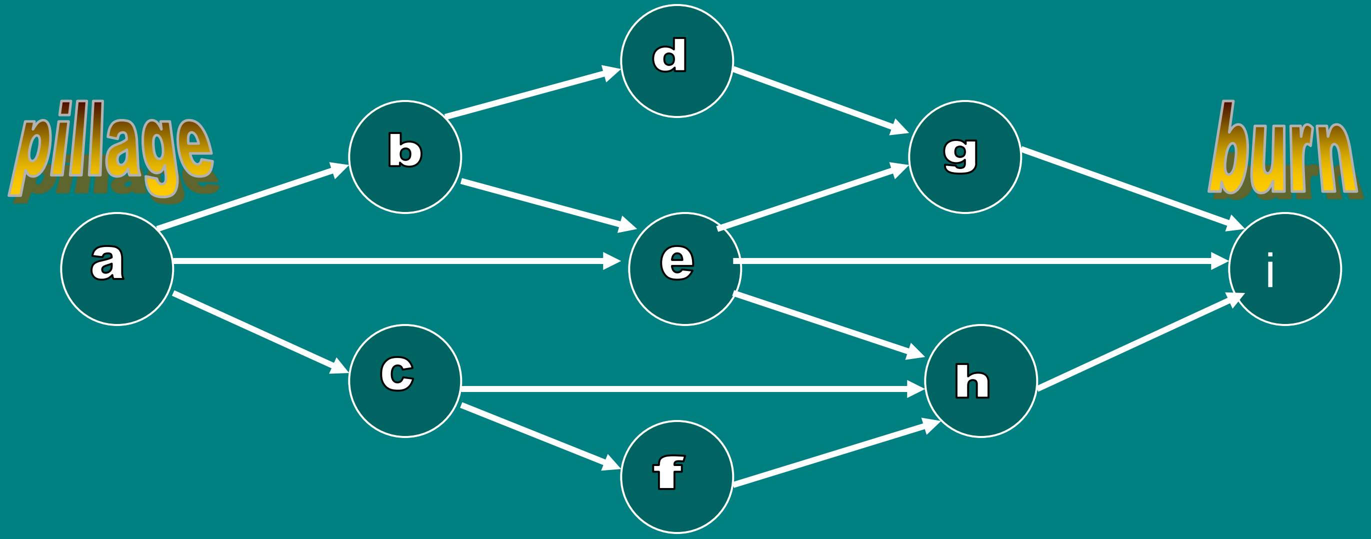
- One way to find a topological sort is to consider the *in-degrees* of the vertices. (The number of incoming edges is the **in-degree**). Clearly the first vertex in a topological sort must have in-degree zero and every DAG must contain at least one vertex with in-degree zero.

Simple Topological Sort Algorithm

- Repeat the following steps until the graph is empty:
  - Select a vertex that has in-degree zero.
  - Add the vertex to the sort.
  - Delete the vertex and all the edges emanating from it from the graph.

Test Your Knowledge

- Give a topological sort for this graph, it should be evident that more than one solution exists for this problem.


pillage
burn
a
b
c
d
e
f
g
h
i