Relations

Dr. Isaac Griffith

Idaho State University

Outline

The lecture is structured as follows:

- Closures of Relations
- Equivalence Relations
- Partial Orderings

Closures of Relations

CS 1187

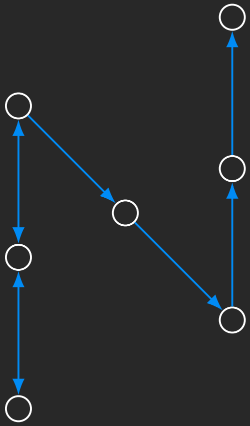
ROAR

Relational Closures

- Three types we will study
  - Reflexive -> Easy
  - Symmetric -> Easy
  - Transitive -> Hard

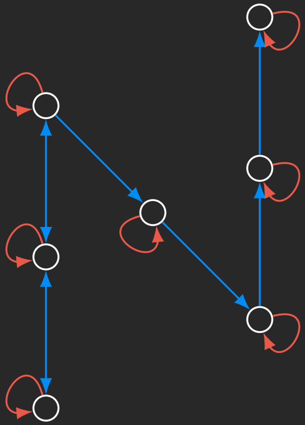
Reflexive Closure

- Consider a relation R depicted in the digraph.
  - Note that it is not reflexive
- We want to add edges to make the relation reflexive
- By adding those edges, we have made a non-reflexive relation *R* into a reflexive relation
- This new relation is called the **reflexive closure** on *R*



Reflexive Closure

- Consider a relation R depicted in the digraph.
  - Note that it is not reflexive
- We want to add edges to make the relation reflexive
- By adding those edges, we have made a non-reflexive relation *R* into a reflexive relation
- This new relation is called the **reflexive closure** on *R*



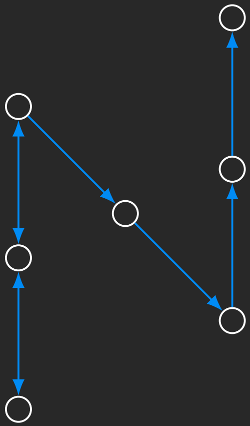
Reflexive Closure

- In order to find the reflexive closure of a relation R, we add a loop at each node that does not have one

- The reflexive closure of R is $R \cup \Delta$
  - Where $\Delta = \{(a, a) \mid a \in R\}$
    - Called the *"Diagonal Relation"*
  - With matrices, we set the diagonal to all 1's

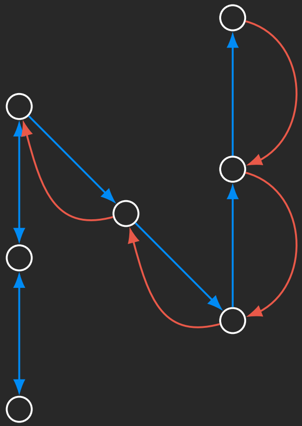
Symmetric Closure

- Consider a relation *R* depicted in the digraph
  - Note that it is not symmetric
- We want to add edge to make the relation symmetric
- By adding those edges, we have made a non-symmetric relation *R* into a symmetric relation
- This new relation is called the **symmetric closure** of *R*



Symmetric Closure

- Consider a relation *R* depicted in the digraph
  - Note that it is not symmetric
- We want to add edge to make the relation symmetric
- By adding those edges, we have made a non-symmetric relation *R* into a symmetric relation
- This new relation is called the **symmetric closure** of *R*



Symmetric Closure

- In order to find the symmetric closure of a relation *R*, we add an edge from *a* to *b*, where there is already an edge from *b* to *a*

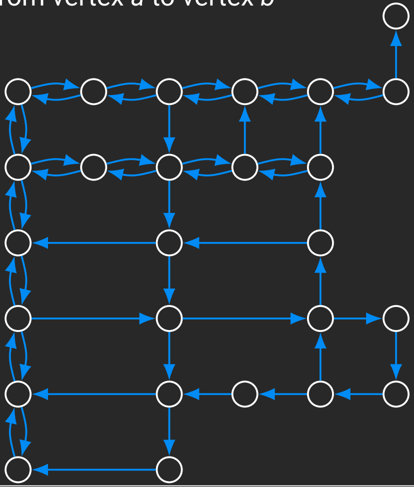
- The symmetric closure of *R* is $R \cup R^{-1}$
  - If $R = \{(a, b) \mid \ldots\}$
  - Then $R^{-1} = \{(b, a) \mid (a, b) \in R\}$

Paths in Directed Graphs

- A *path* is a sequence of connected edges from vertex *a* to vertex *b*

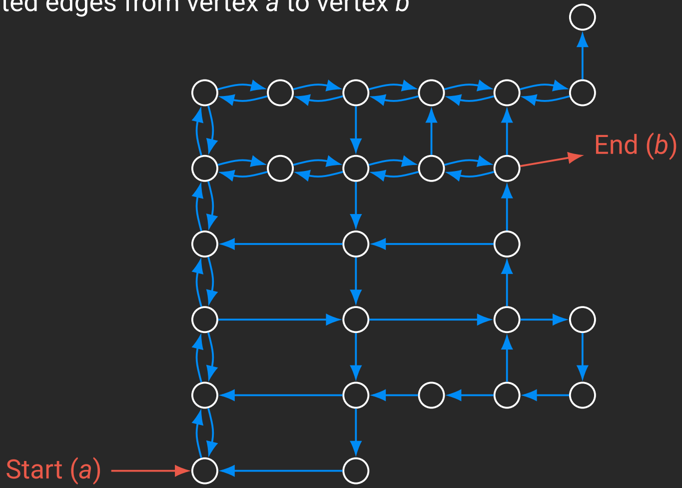
Paths in Directed Graphs

- A *path* is a sequence of connected edges from vertex *a* to vertex *b*



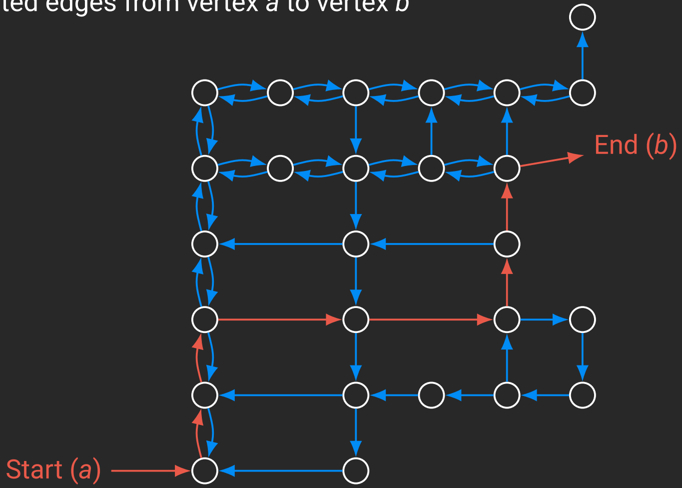
Paths in Directed Graphs

- A *path* is a sequence of connected edges from vertex *a* to vertex *b*



Paths in Directed Graphs

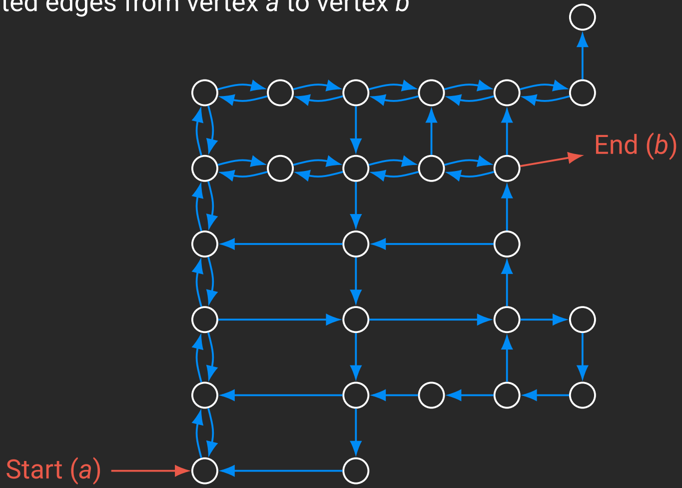
- A *path* is a sequence of connected edges from vertex *a* to vertex *b*



ROAR

Paths in Directed Graphs

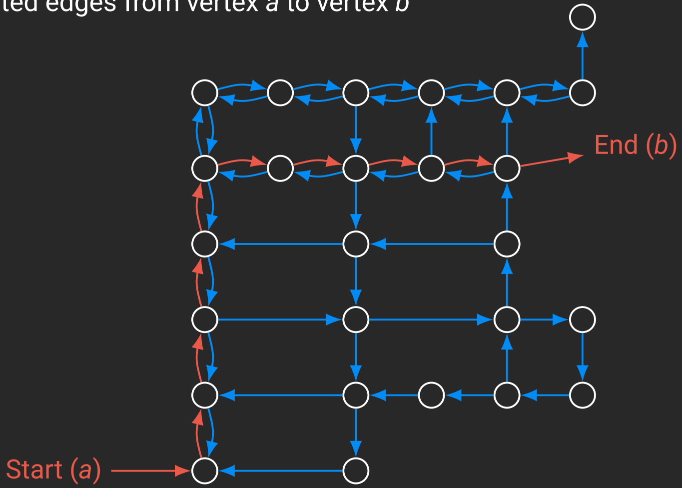
- A *path* is a sequence of connected edges from vertex *a* to vertex *b*



Paths in Directed Graphs

Idaho State University | Computer Science

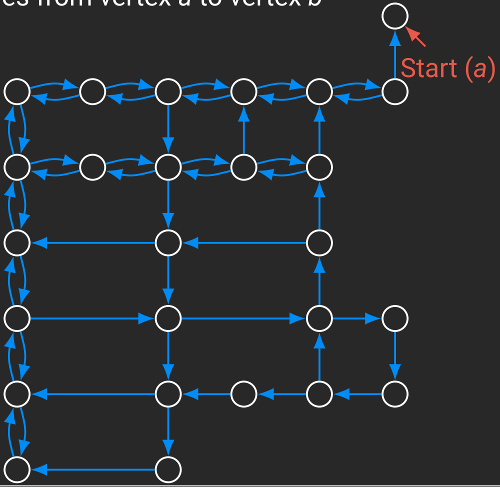
- A *path* is a sequence of connected edges from vertex *a* to vertex *b*



Paths in Directed Graphs

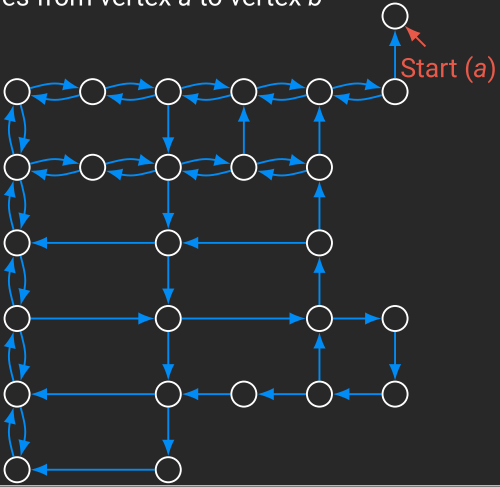
Idaho State University | Computer Science

- A *path* is a sequence of connected edges from vertex *a* to vertex *b*



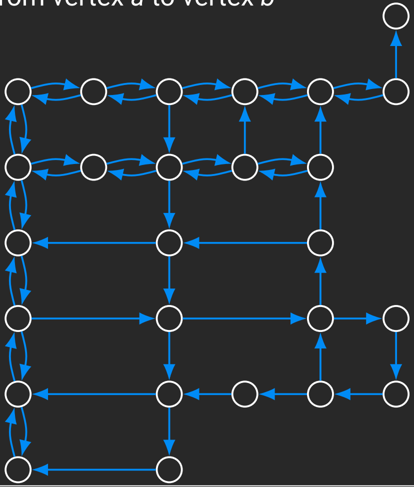
Paths in Directed Graphs

- A *path* is a sequence of connected edges from vertex *a* to vertex *b*
- No path exists from the noted start location



Paths in Directed Graphs

- A *path* is a sequence of connected edges from vertex *a* to vertex *b*



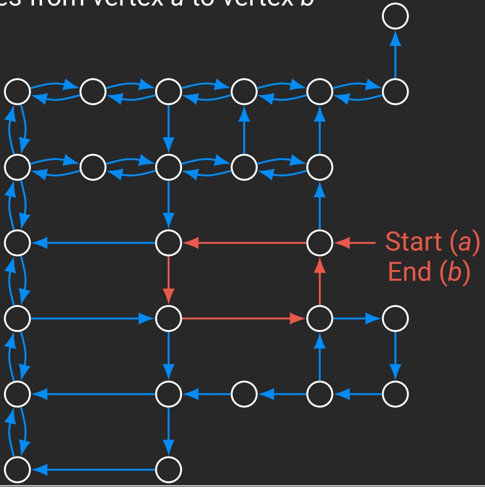
- No path exists from the noted start location
- A path that starts and ends at the same vertex is called a *circuit* or *cycle*

Paths in Directed Graphs

Idaho State University | Computer Science

- A *path* is a sequence of connected edges from vertex *a* to vertex *b*

- No path exists from the noted start location
- A path that starts and ends at the same vertex is called a *circuit* or *cycle*

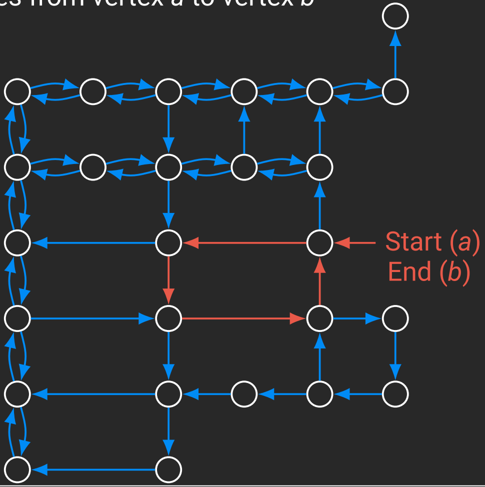


ROAR

Paths in Directed Graphs

- A *path* is a sequence of connected edges from vertex *a* to vertex *b*

- No path exists from the noted start location
- A path that starts and ends at the same vertex is called a *circuit* or *cycle*
  - Must have length ≥ 1



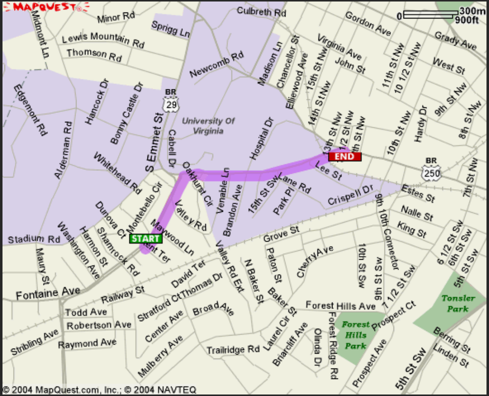
More on Paths…

- The length of a path is the number of **edges** in the path, not the number of nodes

Shortest Paths

Idaho State University | Computer Science

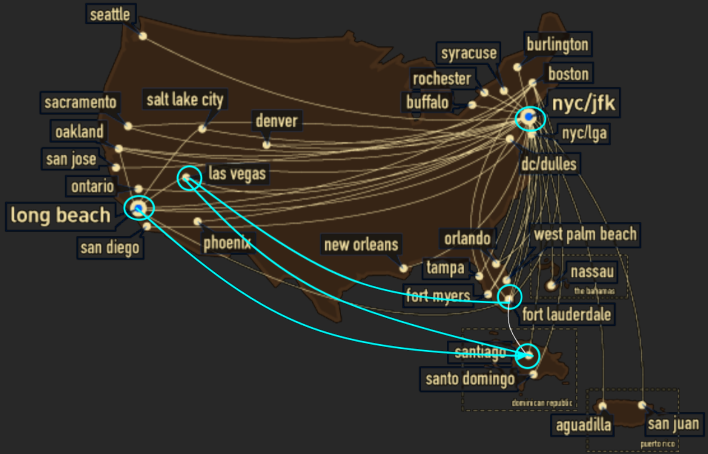
- What is really needed in most applications is finding the shortest path between two vertices



ROAR

Transitive Closure

Idaho State University | Computer Science



The Transitive closure would contain edges between all nodes reachable by a path of any length.

Transitive Closure

Idaho State University | Computer Science

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure

ROAR

Transitive Closure

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:

Transitive Closure

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*

Transitive Closure

Idaho State University | Computer Science

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*

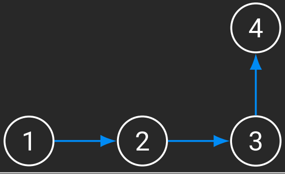
$$R = \{(1,2),(2,3),(3,4)\}$$

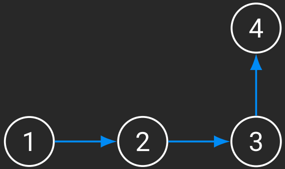
ROAR

Transitive Closure

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*

$$R = \{(1,2),(2,3),(3,4)\}$$

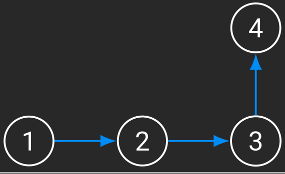




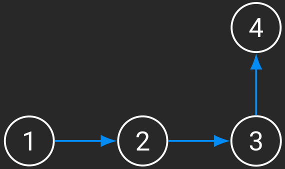
Transitive Closure

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*

$$R = \{(1,2),(2,3),(3,4)\}$$



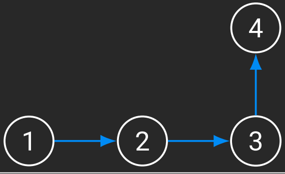
$$(1,2)\ \&\ (2,3) \Rightarrow (1,3)$$



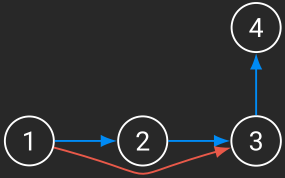
Transitive Closure

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*

$$R = \{(1,2),(2,3),(3,4)\}$$



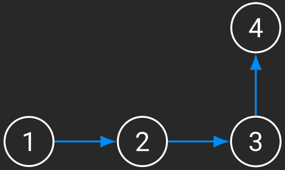
$$(1,2)\ \&\ (2,3) \Rightarrow (1,3)$$



Transitive Closure

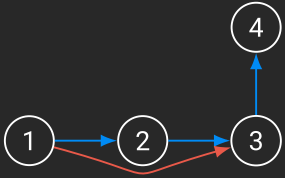
- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:
    - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*

$$R = \{(1,2),(2,3),(3,4)\}$$



$$(1,2)\ \&\ (2,3) \Rightarrow (1,3)$$

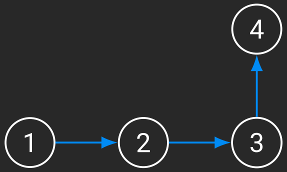
$$(2,3)\ \&\ (3,4) \Rightarrow (2,4)$$



Transitive Closure

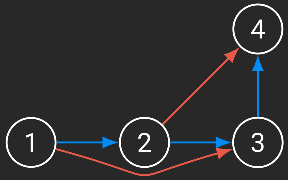
- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*

$$R = \{(1,2),(2,3),(3,4)\}$$



$$(1,2)\ \&\ (2,3) \Rightarrow (1,3)$$

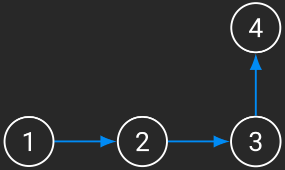
$$(2,3)\ \&\ (3,4) \Rightarrow (2,4)$$



Transitive Closure

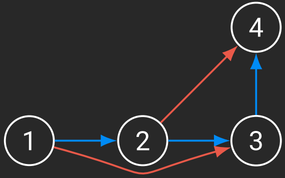
- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*
- But there is a path from 1 to 4 with no edge!

$$R = \{(1,2),(2,3),(3,4)\}$$



$$(1,2)\ \&\ (2,3) \Rightarrow (1,3)$$

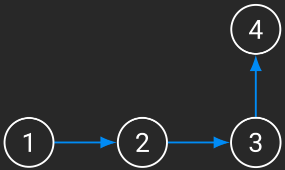
$$(2,3)\ \&\ (3,4) \Rightarrow (2,4)$$



Transitive Closure

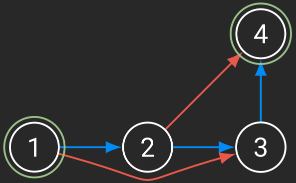
- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- First take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*
- But there is a path from 1 to 4 with no edge!

$$R = \{(1,2),(2,3),(3,4)\}$$



$$(1,2)\ \&\ (2,3) \Rightarrow (1,3)$$

$$(2,3)\ \&\ (3,4) \Rightarrow (2,4)$$



Transitive Closure

Idaho State University | Computer Science

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure

ROAR

Transitive Closure

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- Second take of a definition:

Transitive Closure

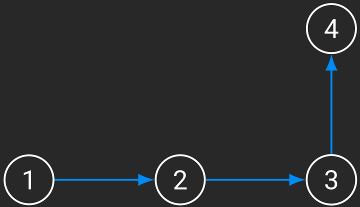
- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- Second take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*

Transitive Closure

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- Second take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*
  - Repeat this step until no new edges are added to the relation

Transitive Closure

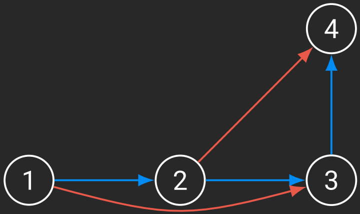
- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- Second take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*
  - Repeat this step until no new edges are added to the relation



Transitive Closure

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- Second take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*
  - Repeat this step until no new edges are added to the relation

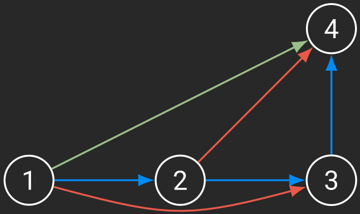
- **FireOpal** means added on the first repeat



Transitive Closure

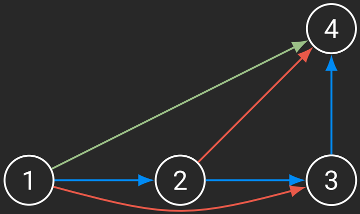
- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- Second take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*
  - Repeat this step until no new edges are added to the relation

- **FireOpal** means added on the first repeat
- **Olivine** means added on the second repeat



Transitive Closure

- Informal definition: If there is a path from *a* to *b*, then there should be an edge from *a* to *b* in the transitive closure
- Second take of a definition:
  - In order to find the transitive closure of a relation *R*, we add an edge from *a* to *c*, when there are edges from *a* to *b* and *b* to *c*
  - Repeat this step until no new edges are added to the relation
- We will study different algorithms for determining the transitive closure
  - **FireOpal** means added on the first repeat
  - **Olivine** means added on the second repeat



6 Degrees of Separation

- The idea that everybody in the world is connected by six degrees of separation
  - Where 1 degree of separation means you know (or have met) somebody else
- Let R be a relation on the set of all people in the world
  - $(a, b) \in R$ if person a has met person b
- So six degrees of separation for *any* two people a and g means
  - (a, b), (b, c), (c, d), (d, e), (e, f), (f, g) are all in R
- Or, $(a, g) \in R^6$

Connectivity Relation

- R contains edges between all the nodes reachable via 1 edge
- $R \circ R = R^2$ contains edges between nodes that are reachable via 2 edges in R
- $R^2 \circ R = R^3$ contains edges between nodes that are reachable via 3 edges in R
- $R^n =$ contains edges between nodes that are reachable via n edges in R
- R^* contains edges between nodes that are reachable via any number of edges (i.e., via any path) in R
  - Rephrased: R^* contains all the edges between nodes a and b when there is a path of length at least 1 between a and b in R
- R^* is the transitive closure of R
  - The definition of a transitive closure is that there are edges between any nodes (a, b) that contain a path between them.

Star Closure

- R^* is the star closure of relation R, and it is defined as

$$R^* = \bigcup_{k=1}^{\infty} R^k$$

- **Definition:** The transitive closure of a relation R, $t(R)$, is the smallest transitive relation containing R.

- **Theorem:** $t(R) = R^*$

Finding the Transitive Closure

Idaho State University | Computer Science

- Let $\mathbf{M}_R$ be the zero-one matrix of the relation R on a set with n elements. Then the zero-one matrix of the transitive closure R^* is:

$$\mathbf{M}_{R^*} = \mathbf{M}_R \vee \mathbf{M}_R^{[2]} \vee \mathbf{M}_R^{[3]} \vee \ldots \vee \mathbf{M}_R^{[n]}$$

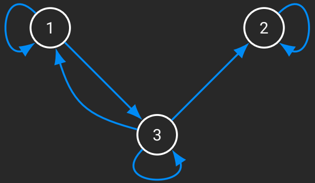
Where:

- $\mathbf{M}_R$ - Nodes reachable with one application of the relation
- $\mathbf{M}_R^{[2]}$ - Nodes reachable with two applications of the relation
- $\mathbf{M}_R^{[n]}$ - Nodes reachable with n applications of the relation

Example

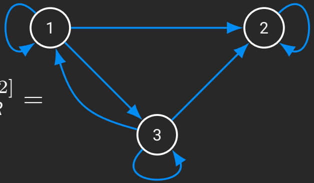
- Find the zero-one matrix of the transitive closure of the relation R given by:

$$\mathbf{M}_R = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$



$$\mathbf{M}_{R^*} = \mathbf{M}_R \vee \mathbf{M}_R^{[2]} \mathbf{M}_R^{[3]}$$

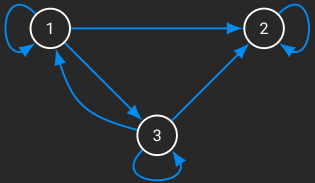
$$\mathbf{M}_R^{[2]} =$$



$$\mathbf{M}_R^{[2]} = \mathbf{M}_R \odot \mathbf{M}_R = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Example

$$\mathbf{M}_R^{[3]} = \mathbf{M}_R^{[2]} \odot \mathbf{M}_R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$



$$\mathbf{M}_R^{[3]} = \mathbf{M}_R^{[2]} \odot \mathbf{M}_R = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \vee \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \vee \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Transitive Closure Algorithm

- What we did (or rather, could have done):
  - Compute the next matrix $\mathbf{M}_R^{[i]}$, where $1 \leq i \leq n$
  - Do a Boolean join with the previously computed matrix

- For the example:
  - Compute $\mathbf{M}_R^{[2]} = \mathbf{M}_R \circ \mathbf{M}_R$
  - Join that with $\mathbf{M}_R$ to yield $\mathbf{M}_R \vee \mathbf{M}_R^{[2]}$
  - Compute $\mathbf{M}_R^{[3]} = \mathbf{M}_R^{[2]} \circ \mathbf{M}_R$
  - Join that with $\mathbf{M}_R \vee \mathbf{M}_R^{[2]}$ from above

Transitive Closure Algorithm

procedure TRANSITIVE_CLOSURE($\mathbf{M}_R$: zero-one $n \times n$ matrix)
  $A := \mathbf{M}_R$
  $B := A$
  for $i := 2$ **to** n **do**
    $A := A \odot \mathbf{M}_R$
    $B := B \vee A$
  return B

- What is the time complexity? $O(n^4)$ bit operations due to the product and join operations within the loop

Roy-Warshall Algorithm

- Uses only $O(n^3)$ bit operations

procedure $\textsc{Warshall}(\mathbf{M}_R$: rank-$n$ 0-1 matrix)
  $W := \mathbf{M}_R$
  for $k := 1$ **to** n **do**
    for $i := 1$ **to** n **do**
      for $j := 1$ **to** n **do**
        $w_{ij} := w_{ij} \vee (w_{ik} \vee w_{kj})$
  return W $\triangleright$ represents R^*

- $w_{ij} = 1$ means there is a path from i to j going only through nodes $\leq k$.

- Indices i and j may have index higher than k

Equivalence Relations

CS 1187

Introduction

- Certain combinations of relation properties are very useful

- In this section we will study equivalence relations:
  - A relation that is *reflexive*, *symmetric*, and *transitive*

- In the next section we will study partial ordering:
  - A relation that is *reflexive*, *antisymmetric*, and *transitive*

- The difference is whether the relation is symmetric or antisymmetric

Equivalence Relations

Idaho State University | Computer Science

We can group properties of relations together to define new types of important relations

- **Definition:** A relation R on a set A is an **equivalence relation** iff R is
  - reflexive
  - symmetric
  - transitive
- Two elements related by an equivalence relation are called **equivalent**
- **Example:** Consider relation $R = \{(a, b) \mid len(a) = len(b)\}$, where $len(a)$ means the length of string a
  - It is reflexive: $len(a) = len(a)$
  - It is symmetric: if $len(a) = len(b)$, then $len(b) = len(a)$
  - It is transitive: if $len(a) = len(b)$ and $len(b) = len(c)$, then $len(a) = len(c)$
  - Thus, R is an equivalence relation

Equivalence Relation Example

Idaho State University | Computer Science

- Consider the relation $R = \{(a, b) \mid a \equiv b \pmod{m}\}$
  - Remember that this means that $m \mid a - b$
  - Called "congruence modulo m"
- Is it reflexive: $(a, a) \in R$ means that $m \mid a - a$
  - $a - a = 0$, which is divisible by m
- Is it symmetric: if $(a, b) \in R$ then $(b, a) \in R$
  - (a, b) means that $m \mid a - b$
  - Or that $km = a - b$. Negating that, we get $b - a = -km$
  - Thus, $m \mid b - a$, so $(b, a) \in R$

ROAR

Equivalence Relation Example

- Consider the relation $R = \{(a,b) \mid a \equiv b(\mod m)\}$
  - Remember that this means that $m \mid a - b$
  - Called "congruence modulo m"
- Is it transitive: if $(a,b) \in R$ and $(b,c) \in R$ then $(a,c) \in R$
  - (a,b) means $m \mid a - b$, or that $km = a - b$
  - (b,c) means $m \mid b - c$, or that $lm = b - c$
  - (a,c) means that $m \mid a - c$, or that $nm = a - c$
  - Adding these two, we get $km + lm = (a - b) + (b - c)$
  - Or $(k + l)\, m = a - c$
  - Thus, m divides $a - c$, where $n = k + l$
- Thus, congruence modulo m is an equivalence relation

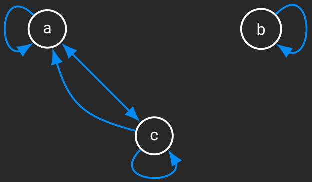
Equivalence Classes

- An **equivalence class** of an element x:
  - $[x] = \{y \mid (x, y) \in R\}$
  - $[x]$ is the subset of all elements related to $[x]$ by R
  - The element in the bracket is called a **representative** of the equivalence class.
    - We could have chosen any one.
- **Theorem:** Let R be an equivalence relation on A. Then either

$$[a] = [b] \text{ or } [a] \cap [b] = \varnothing$$

- The number of equivalence classes is called the *rank* of the equivalence relation

Example: Let $A = \{a, b, c\}$ and R be given by the shown digraph, then



$$[a] = \{a, c\},\ [b] = \{b\},\ [c] = \{a, c\}$$

rank = 2

Equivalence Classes

- Consider the relation $R = \{(a,b) \mid a \mod 2 = b \mod 2\}$ on the set of integers
  - Thus, all the even numbers are related to each other
  - As are the odd numbers
- The even numbers form an equivalence class
  - As do the odd numbers
- The equivalence class for the even numbers is denoted by $[2]$ (or $[4]$, or $[784]$, etc.)
  - $[2] = \{\ldots, -4, -2, 0, 2, -4, \ldots\}$
  - 2 is *representative* of its equivalence class
- There are only 2 equivalence classes formed by this equivalence relation

Equivalence Classes

- Consider the relation: $R = \{(a, b) \mid a = b \vee a = -b\}$
  - Thus, every number is related to additive inverse

- The equivalence class for an integer a:
  - $[7] = \{7, -7\}$
  - $[0] = \{0\}$
  - $[a] = \{a, -a\}$

- There are an infinite number of equivalence classes formed by this equivalence relation

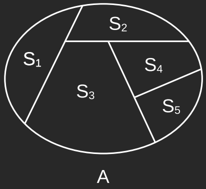
Equivalence Class and Partitions

- **Theorem:** Let R be an equivalence relation on a set A. The equivalence classes of R **partition** the set A into disjoint nonempty subsets whose union is the entire set. This partition is denoted A/R and called
  - The *quotient set*, or
  - the *partition of A induced by R*, or
  - *A modulo R*

ROAR

Equivalence Class and Partitions

- **Definition:** Let $S_1, S_2, \ldots, S_n$ be a collection of subsets of a set A. Then the collection forms a **partition** of A if the subsets are nonempty, disjoint and exhaust A
  - $S_1 \neq \varnothing$
  - $S_i \cap S_j = \varnothing$ if $i \neq j$
  - $\bigcup S_i = A$

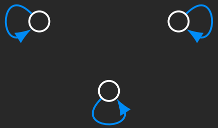


- Note that $\{\{\}, \{1,3\}, \{2\}\}$ is not a partition of $\{1,2,3\}$ (it contains the empty set)
- Note that $\{\{1\}, \{2\}\}$ is not a partition of $\{1,2,3\}$ as none of blocks contain 3

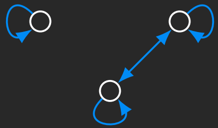
Equivalence Relations and Digraphs

- It is easy to recognize equivalence relations using digraphs:
  - The equivalence class of a particular element forms a universal relation (contains all possible edges) between the elements in the equivalence class
  - The (sub)digraph representing the subset is called a **complete** (sub)digraph, since all edges are present

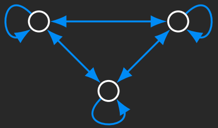
- **Example:** All possible equivalence relations on a set *A* with 3 elements



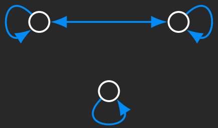
rank = 3



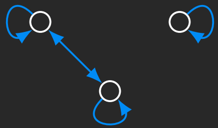
rank = 2



rank = 1



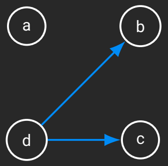
rank = 2



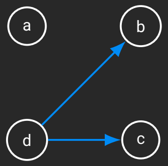
rank = 2

Induced Equivalence Relations

- **Theorem:** If R_1 and R_2 are equivalence relations on A, then $R_1 \cap R_2$ is an equivalence relation on A
- **Definition:** Lt R be a relation on A. Then the reflexive, symmetric, transitive closure of R, $tsr(R)$, is an equivalence relation on A, called the **equivalence relation induced** by R
- **Example:**



R



$tsr(R)$

rank = 2

$$A = [a] \cup [b] = \{a\} \cup \{b, c, d\}$$

$$A/R = \{\{a\}, \{b, c, d\}\}$$

- **Theorem:** $tsr(R)$ is an equivalence relation

ROAR

Partial Orderings

CS 1187

ROAR

Introduction

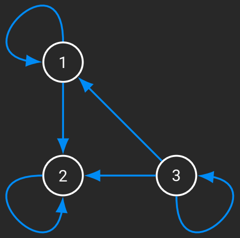
- An equivalence relation is a relation that is reflexive, symmetric, and transitive

- A **partial ordering** (or **partial order**) is a relation that is reflexive, *antisymmetric*, and transitive
  - Recall that antisymmetric means that is $(a, b) \in R$, then $(b, a) \notin R$ unless $b = a$
  - Thus, (a, a) is allowed to be in R
  - But, since it's reflexive, all possible (a, a) must be in R

Partially Ordered Set (POSET)

- **Definition:** A relation R on a set S is called a **partial ordering** or **partial order** if it is *reflexive*, *antisymmetric*, and *transitive*. A set S together with a partial ordering R is called a **partially ordered set**, or **poset**, and is denoted by (S, R)

- **Example:** Let $S = \{1, 2, 3\}$ and $R = \{(1,1), (2,2), (3,3), (1,2), (3,1), (3,2)\}$



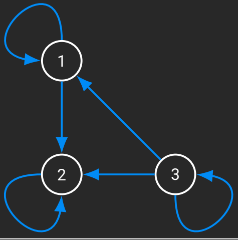
Partially Ordered Set (POSET)

- In a poset the notation $a \preccurlyeq b$ denotes that $(a, b) \in R$

  This notation is used because the "less than or equal to" relation is a paradigm for a partial ordering. (Note that the symbol $\preccurlyeq$ is used to denote the relation in *any* poset, not just the "less than or equals" relation.)

  The notation $a \prec b$ denotes that $a \preccurlyeq b$, but $a \neq b$

- **Example:** Let $S = \{1, 2, 3\}$ and $R = \{(1, 1), (2, 2), (3, 3), (1, 2), (3, 1), (3, 2)\}$



$$\begin{array}{ccc} 2 & \preccurlyeq & 2 \\ 3 & \prec & 2 \end{array}$$

ROAR

Comparable / Incomparable

Idaho State University | Computer Science

- **Definition:** The elements a and b of a poset $(S, \preccurlyeq)$ are called **comparable** if either $a \preccurlyeq b$ or $b \preccurlyeq a$. When a and b are elements of S such that neither $a \preccurlyeq b$ nor $b \preccurlyeq a$, a and b are called **incomparable**

- **Example:** Consider the power set of $\{a, b, c\}$ and the subset relation. $(P(\{a, b, c\}), \subseteq)$

$$\{a, c\} \not\subseteq \{a, b\} \text{ and } \{a, b\} \not\subseteq \{a, c\}$$

So, $\{a, c\}$ and $\{a, \}$ are **incomparable**

ROAR

Totally Ordered

- **Definition:** If $(S, \preccurlyeq)$ is a poset and every two elements of S are comparable, S is called a **totally ordered** or **linearly ordered** set, and $\preccurlyeq$ is called a **total order** or a **linear order**.
  - A totally ordered set is also called a **chain**

ROAR

Greatest/Least Elements

- **Definition:** Let R be a total order on A and suppose $S \subseteq A$. An element $s \in S$ is a **least element** of S iff sRb for every $b \in S$.
  - Note: this implies that (a, s) is not in R for any a unless $a = s$. (There is nothing smaller than s under the order R)

- **Definition:** Let R be a total order on A and suppose $S \subseteq A$. An element $s \in S$ is a **greatest element** of S iff bRs for every $b \in S$
  - Note: this implies that (s, a) is not in R for any a unless $a = s$. (There is nothing larger than s under the order R)

Well-Ordered Set

- **Definition:** $(S, \preccurlyeq)$ is a **well-ordered set** if it is a poset such that $\preccurlyeq$ is a total ordering and such that every nonempty subset of S has a *least element*

- **Example:** Consider the ordered pairs of positive integers, $\mathbb{Z}^+ \times \mathbb{Z}^+$ where $(a_1, a_2) \preccurlyeq (b_1, b_2)$ if $a_1 < b_1$, or if $a_1 = b_1$ and $a_2 \leq b_2$

Examples

- **Example:** $(\mathbb{Z}, \leq)$
  - Is a total ordered poset (every element is comparable to every other element)
  - It has no least element
  - Thus, it is not a well-ordered set

- **Example:** $(S, \leq)$ where $S = \{1, 2, 3, 4, 5\}$
  - Is a total ordered poset (every element is comparable to every other element)
  - Has a least element (1)
  - Thus, it is a well-ordered set

Lexicographic Order

- **Definition:** This ordering is called *lexicographic* because it is the way that words are ordered in the dictionary

- Given two posets (A_1, R_1) and (A_2, R_2) we construct an *induced* partial order R on $A_1 \times A_2$: $(x_1, y_1)\, R(x_2, y_2)$ iff
  - $x_1 R_1 x_2$, or
  - $x_1 = x_2$ and $y_1 R_2 y_2$

- **Example:** Let $A_1 = A_2 = \mathbb{Z}^+$ and $R_1 = R_2 =$ 'divides', then
  - $(2, 4)\, R(2, 8)$ since $x_1 = x_2$ and $y_1 R_2 y_2$
  - $(2, 4)$ is not related under R to $(2, 6)$ since $x_1 = x_2$ but 4 does not divide 6
  - $(2, 4)\, R(4, 5)$ since $x_1 R x_2$. (Note that 4 is not related to 5)

Example

Example: Let $\sum$ be a finite set and suppose R is a partial order relation defined on $\sum$. Define a relation $\preccurlyeq$ on $\sum^*$, the set of all strings over $\sum$, as follows:

- For any positive integers m and n and $a_1a_2\ldots a_m$ and $b_1b_2\ldots b_n$ in $\sum^*$
    1. If $m \leq n$ and $a_i = b_i$ for all $i = 1, 2, \ldots, m$, then
    $$a_1a_2\ldots a_m \preccurlyeq b_1b_2\ldots b_n$$
    2. If for some integer k with $k \leq m$, $k \leq n$, and $k \geq 1$, $a_i = b_i$ for all $i = 1, 2, \ldots, k-1$, and a_kRb_k but $a_k \neq b_k$, then
    $$a_1a_2\ldots a_m \preccurlyeq b_1b_2\ldots b_n$$
    3. If ϵ is the null string and s is any string in $\sum^*$ then $\epsilon \preccurlyeq s$.

ROAR

Well-Ordered Induction

Idaho State University | Computer Science

Principle of Well-Ordered Induction:

- Suppose that S is a well-ordered set. Then $P(x)$ is true for all $x \in S$, if:

  BASIS STEP: $P(x_0)$ is true for the least element of S, and

  INDUCTIVE STEP: For every $y \in S$ if $P(x)$ is true for all $x \prec y$, then $P(y)$ is true

Hasse Diagrams

Idaho State University | Computer Science

- Given any partial order relation defined on a finite set, it is possible to draw the directed graph so that all of these properties are satisfied.

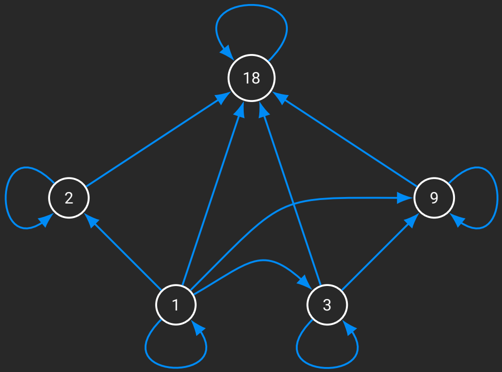
- This makes it possible to associate a somewhat simpler graph, called a **Hasse diagram**, with a partial order relation defined on a finite set.

- Start with a directed graph of the relation in which all arrows point upward. Then eliminate:
  1. The loops at all the vertices
  2. All arrows whose existence is implied by the transitive property
  3. The direction indicators on the arrows

Example

Example: Let $A = \{1, 2, 3, 9, 19\}$ and consider the "divides" relation on A

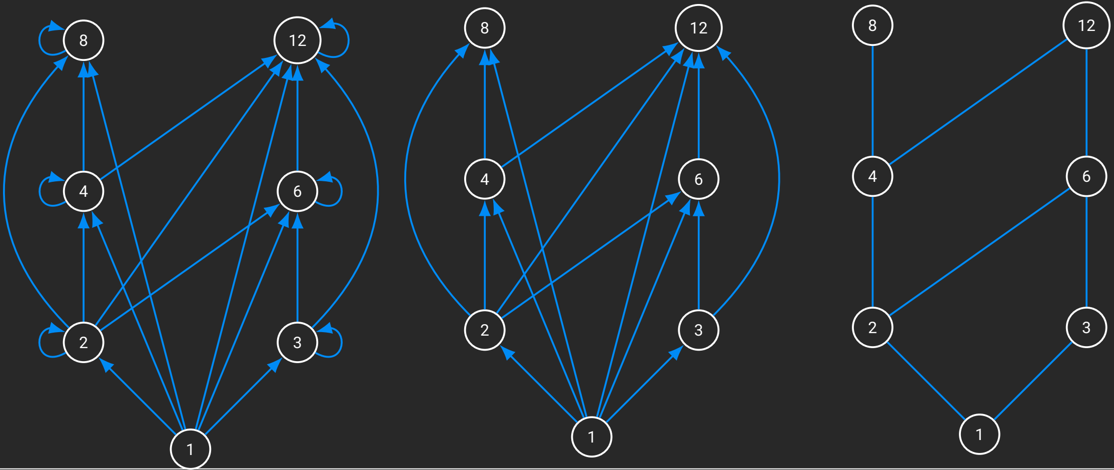
$\forall a, b \in A, a \mid b \leftrightarrow b = ka$ for some integer k



- Eliminate the loops at all the vertices
- Elminiate all arrows whose existence is implied by the transitive property
- Eliminate the direction indicators on the arrows

Hasse Diagram

- For the poset $(\{1, 2, 3, 4, 6, 8, 12\}, |)$

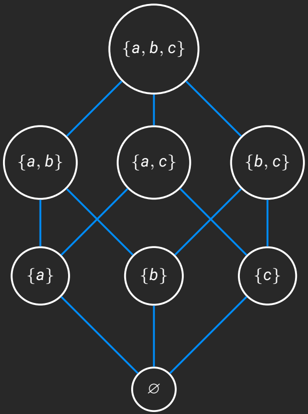


Hasse Diagram

- **Example:** Construct the Hasse diagram of $(P(\{a, b, c\}), \subseteq)$

The elements of $P(\{a, b, c\})$ are:

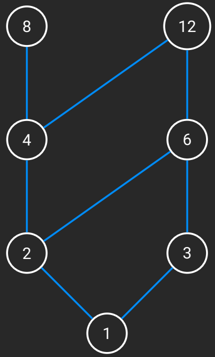
- $\varnothing$
- $\{a\}$, $\{b\}$, $\{c\}$
- $\{a, b\}$, $\{a, c\}$, $\{b, c\}$
- $\{a, b, c\}$



Maximal and Minimal Elements

- **Definition:** a is a **maximal** in the poset $(S, \preccurlyeq)$ if there is no $b \in S$ such that $a \prec b$.

- **Definition:** a is a **minimal** in the poset $(S, \preccurlyeq)$ if there is no element $b \in S$ such that $b \prec a$
  - Note: it is possible to have multiple minimals and maximals



GLEs and ULBs

- **Definition:** a is the **greatest element** in the poset $(S, \preccurlyeq)$ if $b \preccurlyeq a$ for all $b \in S$.
- **Definition:** a is the **least element** in the poset $(S, \preccurlyeq)$ if $a \preccurlyeq b$ for all $b \in S$.
- Sometimes it is possible to find an element that is greater than all the elements in a subset A of a poset $(S, \preccurlyeq)$.
- **Definition:** If u is an element of S such that $a \preccurlyeq u$ for all elements $a \in A$, then u is called an **upper bound** of A
- There may also be an element less than all the elements in A
- **Definition:** If l is an element of S such that $l \preccurlyeq a$ for all elements $a \in A$, then l is called a **lower bound** of A

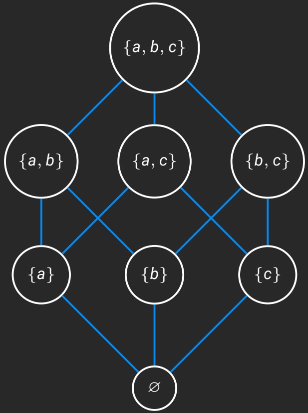
LUB/GLB

- **Definition:** The element x is called the **least upper bound** (lub) of the subset A if x is an upper bound that is less than every other upper bound of A

- **Definition:** The element y is called the **greatest lower bound** (glb) of the subset A if y is a lower bound of A and $z \preccurlyeq y$ whenever z is a lower bound of A.

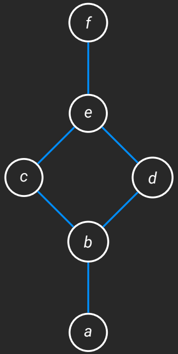
Lattices

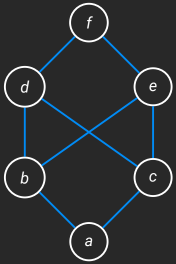
- **Definition:** A partially ordered set in which *every pair* of elements has both a least upper bound and a greatest lower bound is called a **lattice**

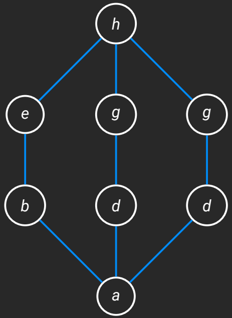


Lattice Example

Example: Determine whether the posets represented by each of the following Hasse diagrams are lattices.





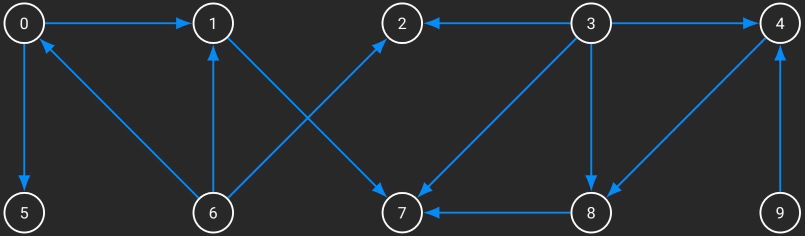


Solution: both the first and third Hasse diagrams are latices, however the second is not since both *b* and *c* do not have least upper bounds.

Topological Sorting

- A total ordering $\preccurlyeq$ is said to be compatible with the partial ordering R if $a \preccurlyeq b$ whenever aRb. Constructing a total ordering from a partial ordering is called **topological sorting**

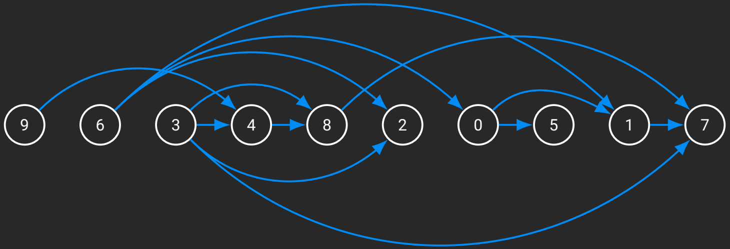
- If there is an edge from v to w, then v precedes w in the sequential listing



Topological Sorting

- A total ordering $\preccurlyeq$ is said to be compatible with the partial ordering R if $a \preccurlyeq b$ whenever aRb. Constructing a total ordering from a partial ordering is called **topological sorting**

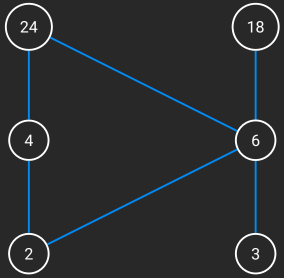
- If there is an edge from v to w, then v precedes w in the sequential listing



ROAR

Example

Example: Consider the set $A = \{2, 3, 4, 6, 18, 24\}$ ordered by the "divides" relation. The Hasse diagram follows:



The ordinary "less than or equal to" relation $\leq$ on this set is a topological sorting for it since for positive integers a and b, if $a \mid b$, then $a \leq b$

Toplogical Sorting

procedure TOPOLOGICALSORT$((S, \preccurlyeq)$: finite poset)

$k := 1$

while $S \neq \varnothing$ **do**

$a_k :=$ a minimal element of S

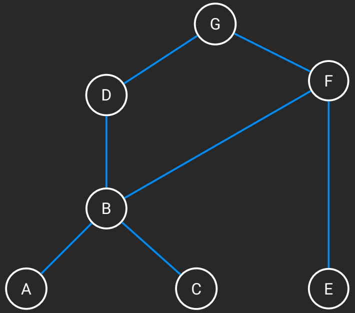
$S := S - \{a_k\}$

$k := k + 1$

return $a_1, a_2, \ldots, a_n$

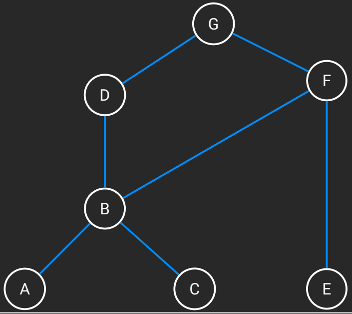
Example

- **Example:** A development project at a computer company requires the completion of seven tasks. Some of these tasks can be started only after other tasks are finished. A partial ordering on tasks is set up by considering task $X \prec$ task Y if task Y cannot be started until task X has been completed.
  - The Hasse diagram for the seven tasks, with respect to this partial ordering is shown below.
  - Find an order in which these tasks can be carried out to complete the project.



Example

- **Example:** A development project at a computer company requires the completion of seven tasks. Some of these tasks can be started only after other tasks are finished. A partial ordering on tasks is set up by considering task $X \prec$ task Y if task Y cannot be started until task X has been completed.
  - The Hasse diagram for the seven tasks, with respect to this partial ordering is shown below.
  - Find an order in which these tasks can be carried out to complete the project.

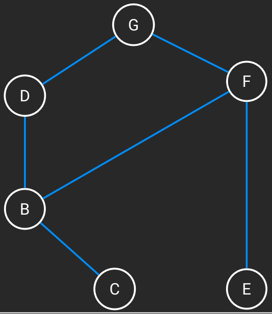


Solution:

ROAR

Example

- **Example:** A development project at a computer company requires the completion of seven tasks. Some of these tasks can be started only after other tasks are finished. A partial ordering on tasks is set up by considering task $X \prec$ task Y if task Y cannot be started until task X has been completed.
  - The Hasse diagram for the seven tasks, with respect to this partial ordering is shown below.
  - Find an order in which these tasks can be carried out to complete the project.

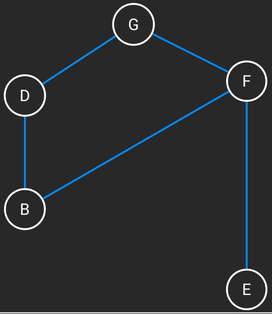


Solution:

- A

Example

- **Example:** A development project at a computer company requires the completion of seven tasks. Some of these tasks can be started only after other tasks are finished. A partial ordering on tasks is set up by considering task $X \prec$ task Y if task Y cannot be started until task X has been completed.
  - The Hasse diagram for the seven tasks, with respect to this partial ordering is shown below.
  - Find an order in which these tasks can be carried out to complete the project.

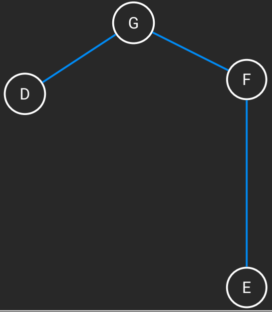


Solution:

- A
- C

Example

- **Example:** A development project at a computer company requires the completion of seven tasks. Some of these tasks can be started only after other tasks are finished. A partial ordering on tasks is set up by considering task $X \prec$ task Y if task Y cannot be started until task X has been completed.
  - The Hasse diagram for the seven tasks, with respect to this partial ordering is shown below.
  - Find an order in which these tasks can be carried out to complete the project.



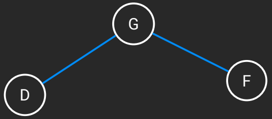
Solution:

- A
- C
- B

ROAR

Example

- **Example:** A development project at a computer company requires the completion of seven tasks. Some of these tasks can be started only after other tasks are finished. A partial ordering on tasks is set up by considering task $X \prec$ task Y if task Y cannot be started until task X has been completed.
  - The Hasse diagram for the seven tasks, with respect to this partial ordering is shown below.
  - Find an order in which these tasks can be carried out to complete the project.

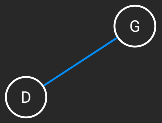


Solution:

- A
- C
- B
- E

Example

- **Example:** A development project at a computer company requires the completion of seven tasks. Some of these tasks can be started only after other tasks are finished. A partial ordering on tasks is set up by considering task $X \prec$ task Y if task Y cannot be started until task X has been completed.
  - The Hasse diagram for the seven tasks, with respect to this partial ordering is shown below.
  - Find an order in which these tasks can be carried out to complete the project.



Solution:

- A
- C
- B
- E
- F

Example

- **Example:** A development project at a computer company requires the completion of seven tasks. Some of these tasks can be started only after other tasks are finished. A partial ordering on tasks is set up by considering task $X \prec$ task Y if task Y cannot be started until task X has been completed.
  - The Hasse diagram for the seven tasks, with respect to this partial ordering is shown below.
  - Find an order in which these tasks can be carried out to complete the project.

Solution:

- A
- C
- B
- E
- F
- D

Example

- **Example:** A development project at a computer company requires the completion of seven tasks. Some of these tasks can be started only after other tasks are finished. A partial ordering on tasks is set up by considering task $X \prec$ task Y if task Y cannot be started until task X has been completed.
  - The Hasse diagram for the seven tasks, with respect to this partial ordering is shown below.
  - Find an order in which these tasks can be carried out to complete the project.

Solution:

- A
- C
- B
- E
- F
- D
- G

$$A \prec C \prec B \prec E \prec F \prec D \prec G$$

Are there any questions?