1 101100001000 101100001000 101100001000 101100001000 101100001000 1

CSCI 2824: Discrete Structures

Lecture 35: Equivalence Relations

Rachel Cox

Department of Computer Science

0 101100001000 101100001000 101100001000 101100001000 101100001000 0

Equivalence Classes

Warmup Example: Consider the relation R on the set $A = \{1, 2, 3, 4\}$ given by
$R = \{(1,1), (1,2), (2,1), (2,2), (3,3), (3,4), (4,3), (4,4)\}$
Is this an equivalence relation?

$\boxed{\text{yes!}}$

practice on your own

Equivalence Classes

Equivalence relations can help to divide a set into smaller groups of equivalent things (we get to decide what it means to be "equivalent")

Such a grouping is called a ***partition***. Each of the individual groups is called an ***equivalence class***.

Let R be an equivalence relation on the set A $(R \subseteq A \times A)$. The set of all elements that are related to element a in A is called the ***equivalence class*** of a. We denote this as:

$$[a]_R = \{\text{set of all elements in } A \text{ related to } a\} = \{s \mid (a, s) \in R\}$$

Equivalence Classes

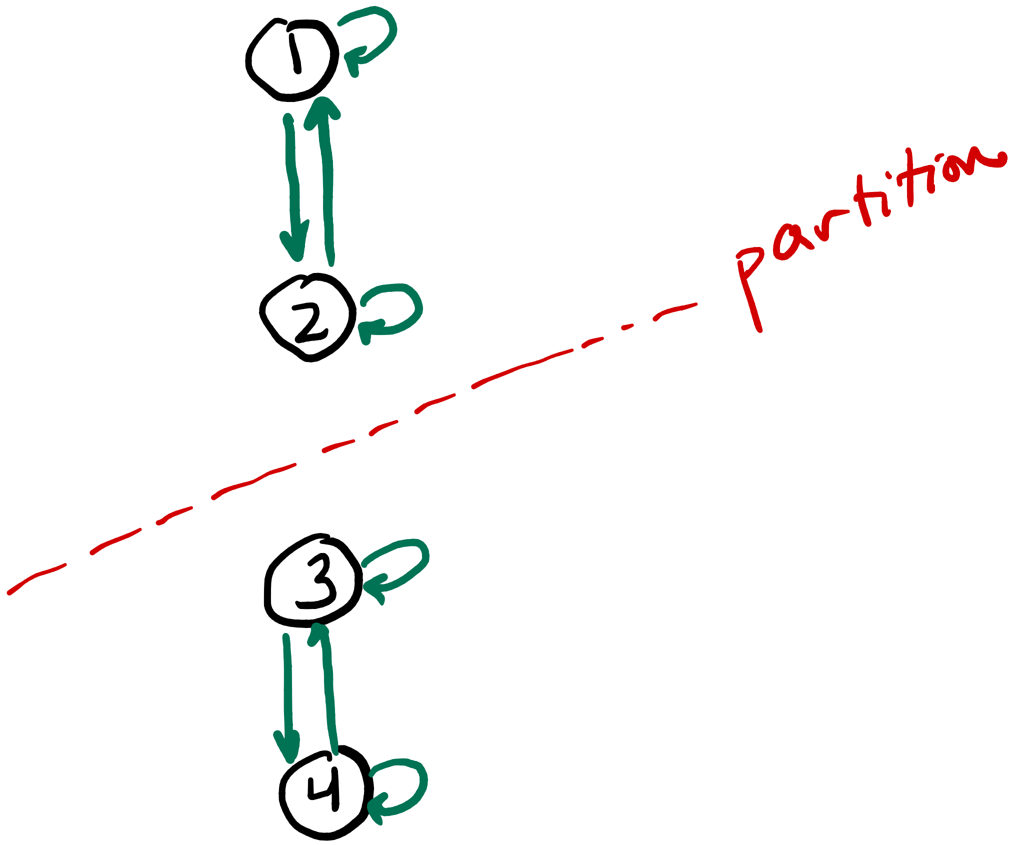
Warmup Example: Consider the relation R on the set $A = \{1, 2, 3, 4\}$ given by
$R = \{(1,1), (1,2), (2,1), (2,2), (3,3), (3,4), (4,3), (4,4)\}$
Is this an equivalence relation? YES!

What things are equivalent to 1?

$$[1]_R = [2]_R = \{1, 2\}$$

What things are equivalent to 3?

$$[3]_R = [4]_R = \{3, 4\}$$



Equivalence Classes

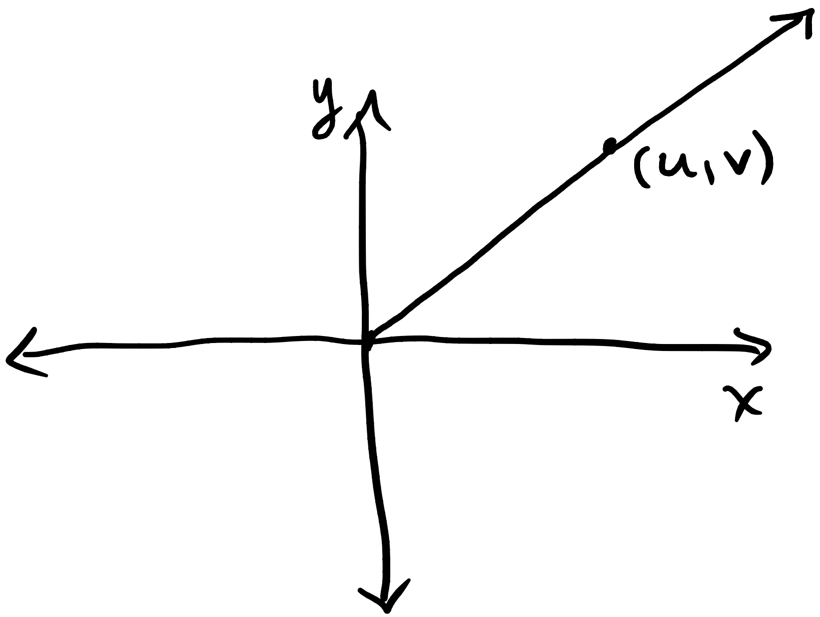
Example: Give an example of an equivalence relation defined on the set of all buildings on campus.

$R_1 = \{ (a,b) \mid a \text{ has same number of floors as } b \}$

The equivalence classes formed by this Relation would be groups of buildings all with the same number of floors.

Equivalence Classes

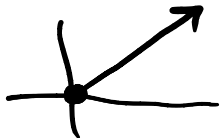
Example: Let A be the set of points in the 2D plane. Rays that start at the origin and go out to infinity is an equivalence relation. What are the equivalence classes?



An equivalence class for (u,v) is the set of all points on the Ray going from the origin through (u,v)

$$\left[(u,v)\right]_R = \{(x,y) \in \mathbb{R}^2 \mid \exists \lambda > 0 \text{ such that } (u,v) = \lambda(x,y)\}$$

e.g. Consider the line $y = x$



Two points: $(1,1), (4,4)$

$(4,4) = 4(1,1)$

Equivalence Classes

Example: The set of all fractional numbers can be partitioned into equivalence classes.

For example: $\frac{1}{2} = \frac{2}{4} = \frac{3}{6} = \frac{4}{8} = \frac{5}{10} = \ldots$

How can we formalize the equivalence relation that induces this partitioning?

$\frac{1}{2} = \frac{2}{4} = \frac{3}{6} = \frac{4}{8} \ldots$

$\frac{1}{5} = \frac{2}{10} \ldots$

$\frac{1}{3} = \frac{3}{9} = \frac{6}{18} \ldots$

$\frac{1}{4} = \frac{4}{16} = \ldots$

Two fractions are equal:

$\frac{a}{b} = \frac{c}{d} \Rightarrow ad = cb$

$R = \{(a,b),(c,d) \mid ad = bc\}$

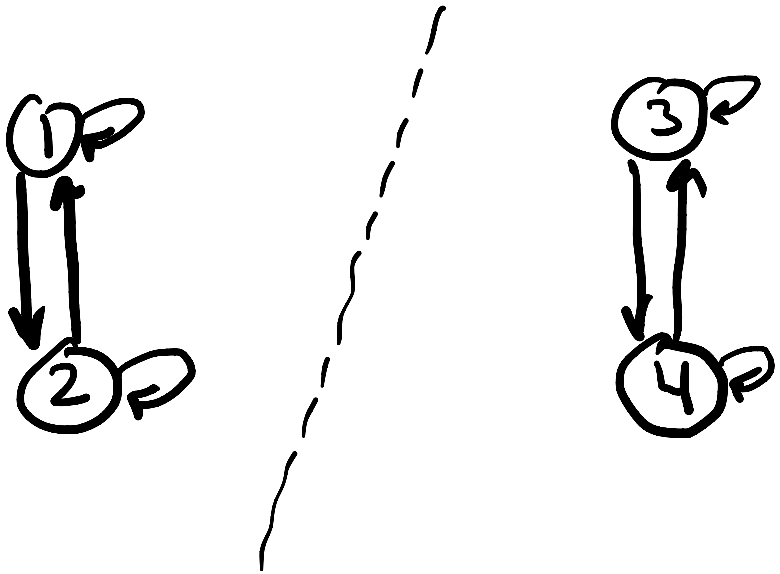
Equivalence Classes

Example: Consider the relation R on the set $A = \{1, 2, 3, 4\}$ given by

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

We showed previously that R is an equivalence relation, with the equivalence classes: $\{1, 2\}$ and $\{3, 4\}$.

What does this look like as a graph?

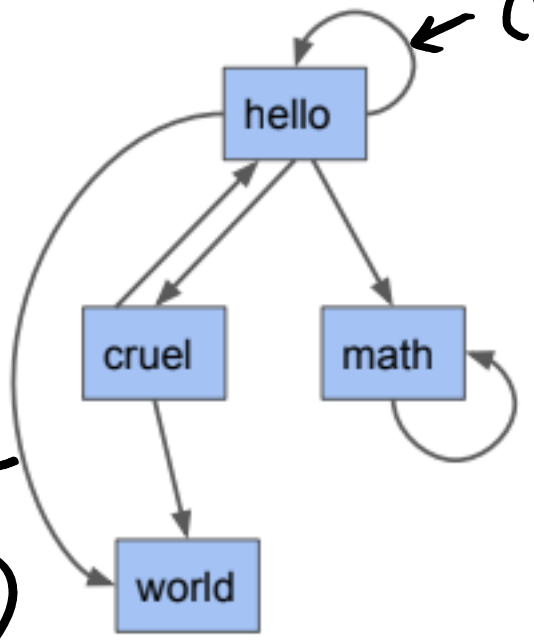


Directed Graphs

Graphs are a way to visualize relations over sets.

A directed graph, or digraph, G consists of a finite set of vertices V and a set of edges $E \subseteq V \times V$, where each edge is represented by a tuple $(u, v) \in E$, where $u, v \in V$.

(hello, hello)



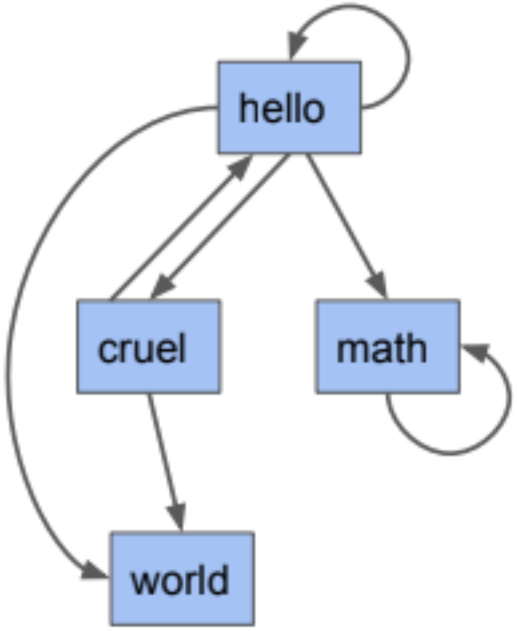
(hello, world)

- Order matters.
- The edges (hello, hello) and (math, math) are called self-loops.

Directed Graphs

Example: Let $V = \{hello, world, math, cruel\}$

Write down the relation E on the set V depicted in this digraph.



$$E = \{(hello, hello), (hello, math), (math, math), (hello, cruel), (cruel, hello), (cruel, world), (hello, world)\}$$

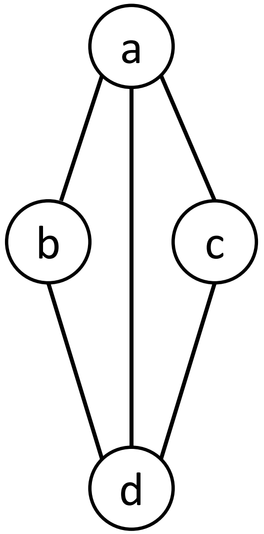
Undirected Graphs

An ***undirected graph*** G consists of a finite set of vertices V and a set of edges $E \subseteq V \times V$, such that the relation E is symmetric.

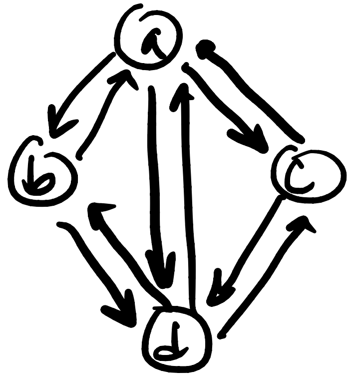
❖ An undirected graph is a special kind of directed graph where the represented edge relation is necessarily symmetric.

Better

Example: $V = \{a, b, c, d\}$ with
E= $\{(a,b),(b,a),(a,c),(c,a),(a,d),(d,a),(b,d),(d,b),(c,d),(d,c)\}$

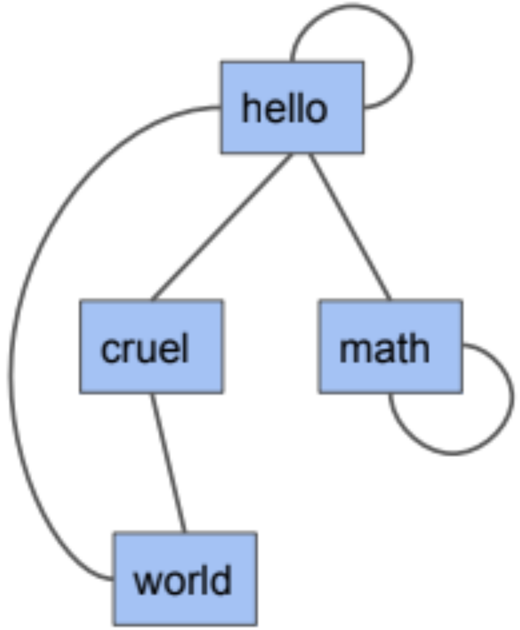


As a directed graph



Undirected Graphs

Example: Let $V = \{hello, world, math, cruel\}$ and let
$E = \{(hello, hello), (hello, cruel), (cruel, hello), (hello, world), (world, hello),$
$(hello, math), (math, hello), (world, cruel), (cruel, world), (math, math)\}$



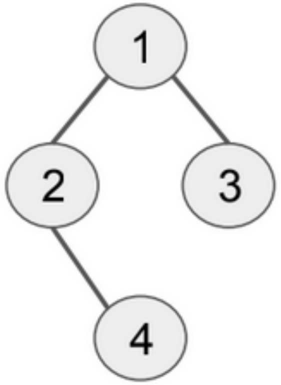
Representing Graphs

Data Structures – Graphs are represented using ***adjacency lists*** or ***adjacency matrices***

Adjacency List:

G = { 1 : [2, 3] , 2 : [1, 4] , 3 : [1] , 4 : [2] }

python



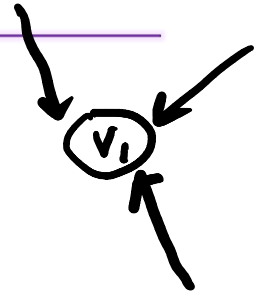
Adjacency Matrix:

$$\begin{array}{ccc} & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} & \\ G = & \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} & \begin{array}{l} \leftarrow \text{vertex 1} \\ \\ \\ \\ \end{array} \end{array}$$

- Each row corresponds to a vertex
- Each column correspond to other vertices: 1 if they're connected, 0 if they aren't

Vertex Degree

Let $G = (V, E)$ be a graph with vertex set V and edge set E.



The set of incoming edges to a vertex $v \in V$ is the set $incoming(v) = \{(u, v) \mid (u, v) \in E\}$

The ***in-degree*** of a vertex v is the number of incoming edges into v. That is:

$$in - degree(v) = |incoming(v)|$$

The set of outgoing edges to a vertex $v \in V$ is the set $outgoing(v) = \{(v, u) \mid (v, u) \in E\}$

The ***out-degree*** of a vertex v is the number of outgoing edges into v. That is:

$$out - degree(v) = |outgoing(v)|$$

Vertex Degree

In an ***undirected graph***, the set of incoming edges to a vertex is the same as the set of outgoing edges, because the edge relation is symmetric.

$$degree(v) = in - degree(v) = out - degree(v)$$

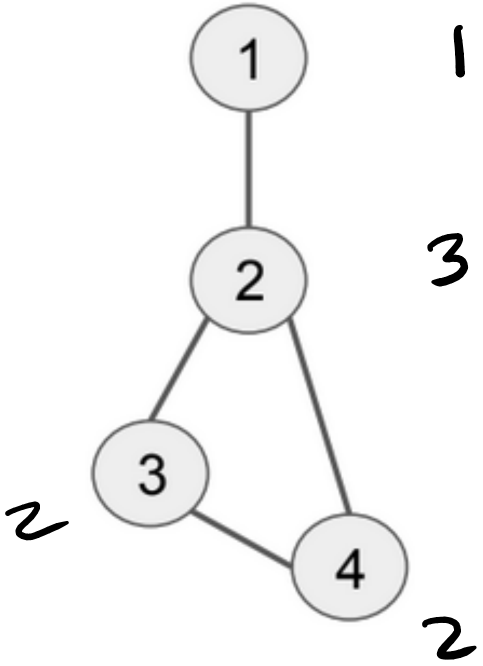
- Let $G = (V, E)$ be an undirected graph without any self-loops.

The ***degree sequence*** of G is a list (usually ordered alphabetically or numerically by vertex name) of the degree of each vertex.

Vertex Degree

vertices: 1, 2, 3, 4

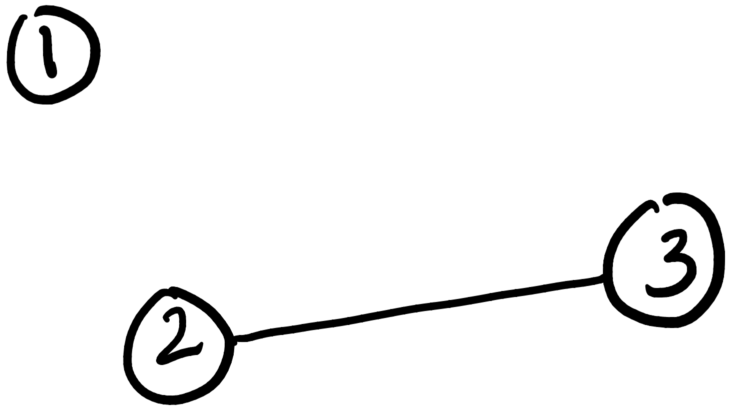
<u>Example</u>: The degree sequence of the graph below is 1, 3, 2, 2



Degree Sequences

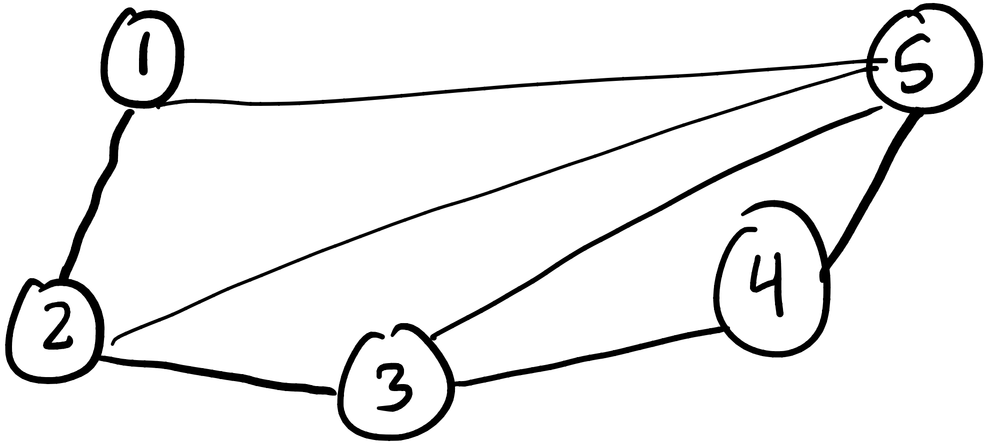
Example: Can you construct a graph G (with no self-loops) with degree sequence 0, 1, 1 ?

yes!



Degree Sequences

Example: Can you construct a graph G (with no self-loops) with degree sequence 1, 2, 3, 4, 5?



no!

not possible

Degree Sequences

<u>The Handshake Theorem</u>: Let $G = (V, E)$ be an undirected graph with no self-loops. Then the sum of each degree in its degree sequence is twice the number of edges. In other words,

$$\sum_{v \in V} degree(v) = 2|E|$$

- Intuition: When computing the degree of a vertex *v*, you count each edge of the form (v, u), where *u* is connected to *v*.

- Later, when computing the degree of the other vertex u, you will count the symmetric edge (u, v), so each edge gets counted exactly twice.

Degree Sequences

<u>The Handshake Theorem</u>: Let $G = (V, E)$ be an undirected graph with no self-loops. Then the sum of each degree in its degree sequence is twice the number of edges. In other words,

$$\sum_{v \in V} degree(v) = 2|E|$$

<u>Corollary 1</u>: The sum of the degree sequence of an undirected graph (with no self-loops) must be even.

<u>Corollary 2</u>: If any vertices have odd degree, then there must be an even number of vertices with odd degree.

Degree Sequences

In/Out degrees for directed graphs:

For a directed graph $G = (V, E)$, $\sum_{v \in V} in - degree(v) = \sum_{v \in V} out - degree(v) = |E|$

- Each directed edge contributes to the in-degree of one vertex and to the out-degree of another vertex.
- This lends credence to the Handshake Theorem if we think of an undirected graph as a directed graph with symmetric edges (each connection is really two connections, one going each way)

- If you've got a set A and an equivalence relation R on A, then R partitions A into equivalence classes.

- Graphs are a nice way to study the structure of sets and networks.

Next time: Walking on Graphs!