CSCI 136:
Data Structures
and
Advanced Programming

Lecture 30

Graphs

Instructor: Dan Barowy

Williams

Topics

Graphs

Your to-dos

1. Read **before Fri**: *Bailey*, Ch. 13.1.
2. Lab 10 (partner lab), **due Tuesday 5/10 by 10pm**.

Announcements

1. **Final exam**: Sunday, May 22, 9:30am in TPL 205.
2. Note that all of the **practice quiz solutions** are on the course website.

Announcements

Suresh Venkatasubramanian (White House; Brown U)

Thursday, May 5 @ 7:30pm*
Bronfman Auditorium – Wachenheim B11
"Machine Readable": The Power and Limits of Algorithms that are Shaping Society
*Talk open to the public. Private reception to follow for Williams students, faculty and staff.

Friday, May 6 @ 2:35pm*
Computer Science Colloquium – Wege TCL 123
On Equity in Access
*Williams students, faculty and staff only.

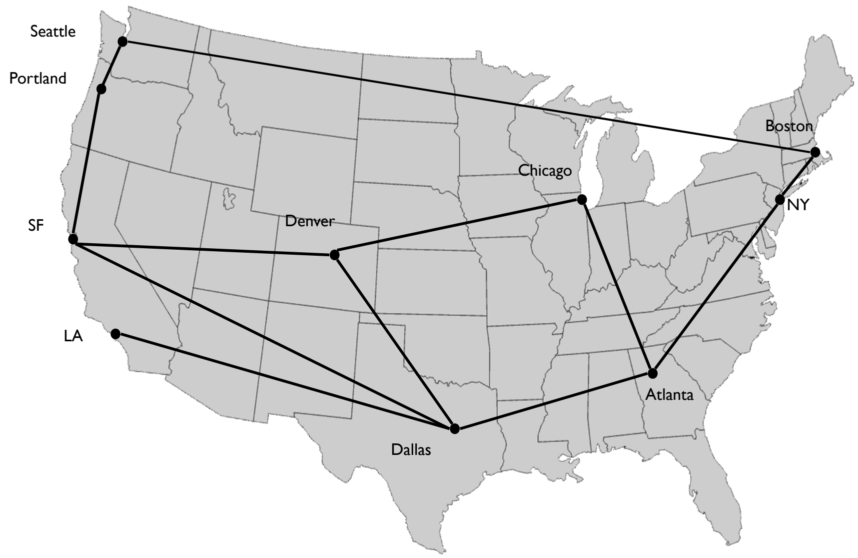
Suresh Venkatasubramanian is a professor in computer science and data science, currently at the White House in the Office of Science and Technology Policy. His background is in theoretical computer science, and he's taken a long and winding path through many areas of data science. For almost the past decade, he's been interested in algorithmic fairness, and more broadly the impact of automated decision-making systems in society.

Graphs

Tons of Applications

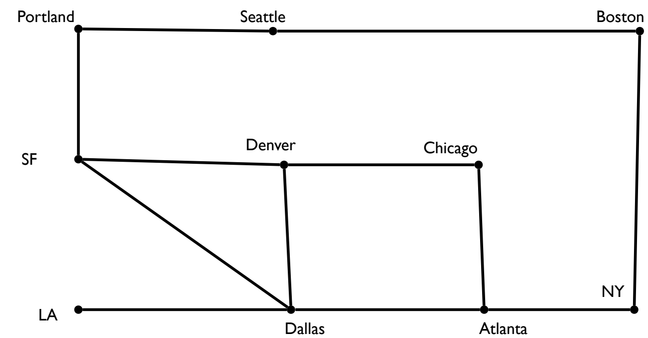
Nodes = subway stops; Edges = track between stops

Tons of Applications



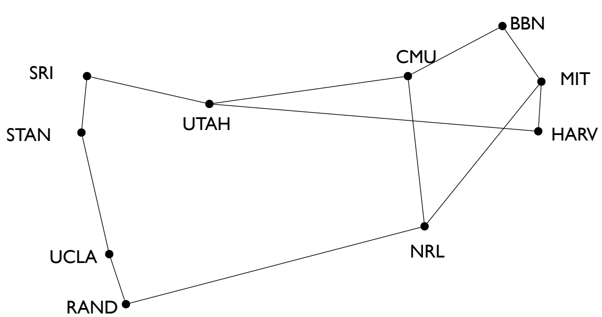
Nodes = cities; Edges = rail lines connecting cities

Tons of Applications



Note: A connection in a graph matters, but not the location of a node.

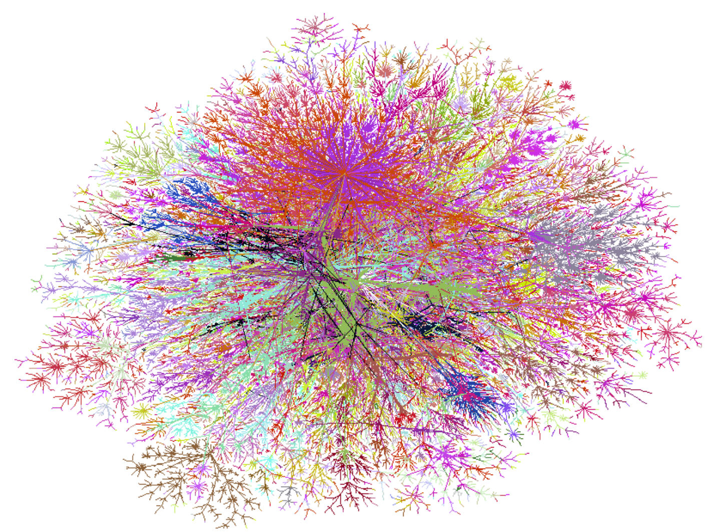
Tons of Applications



Any guesses as to what this is?

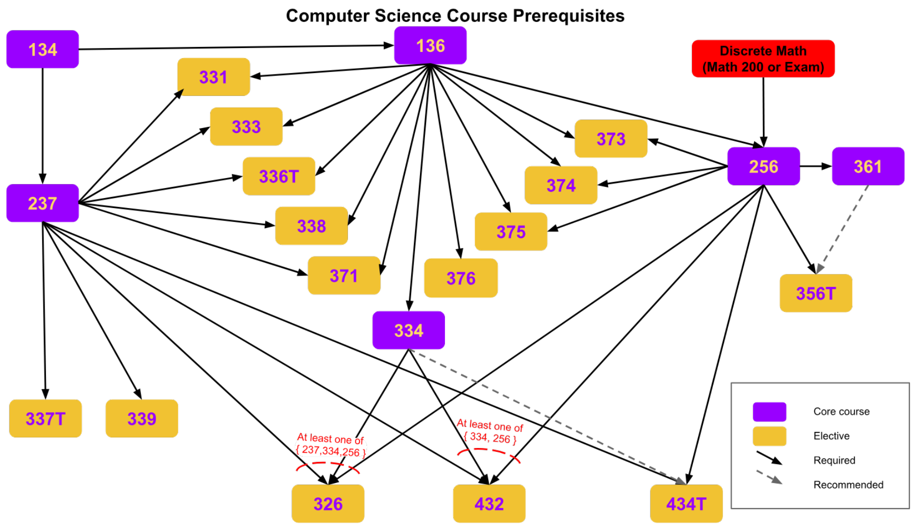
(The Internet, circa 1972.)

Tons of Applications

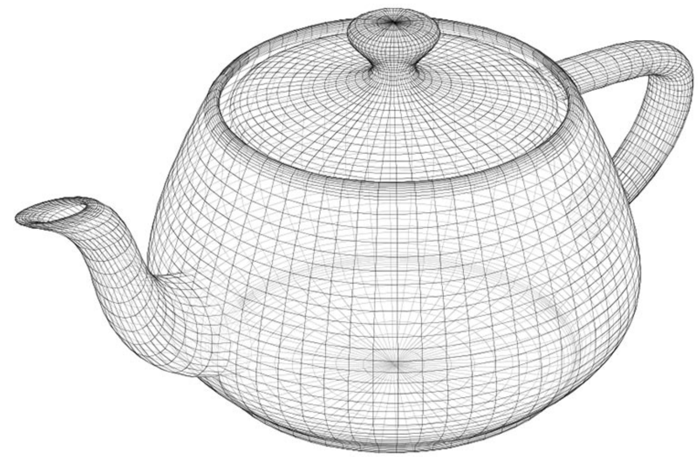


(The Internet, circa 1998.)

Tons of Applications

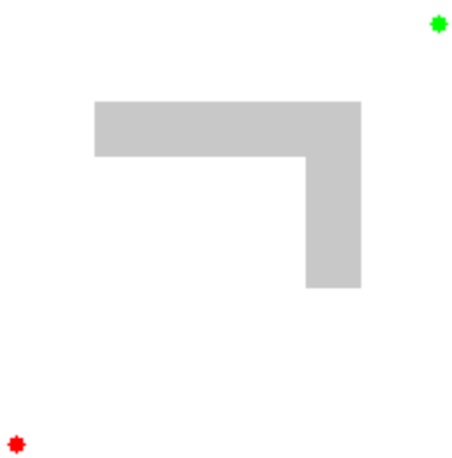


Tons of Applications



A “wireframe” model

Dijkstra’s Algorithm



Undirected graph ADT

An **undirected graph** **G** is an abstract data type that consists of two sets:

- a set **V** of **vertices** (or **nodes**), and
- a set **E** of **undirected edges**.

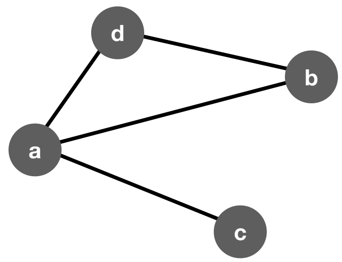
A graph can be used to represent any structure in which pairs of elements are, in some sense, “related.”

In an undirected graph, data can be associated either with a vertex, an edge, or both.

Example: vertex data = city; edge data = distance.

Undirected edges make sense here because the distance from Williamstown to Boston is the same as the distance from Boston to Williamstown.

Undirected graph



G =(V, E)

Directed graph ADT

A **directed graph** **G** is an abstract data type that consists of two sets:

- a set **V** of **vertices** (or **nodes**), and
- a set **E** of **directed edges**.

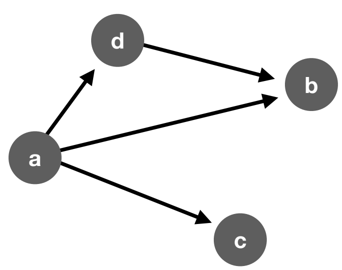
A directed graph can be used to represent any structure in which pairs of elements are “one-way related.”

In a directed graph, data can be associated either with a vertex, an edge, or both.

Example: vertex data = people; edge data = “loves”.

Directed edges make sense here because… unrequited love. See (countless) examples from popular culture.

Directed graph



G =(V, E)

Walking a graph

A **walk from u to v** in a graph **G = (V, E)** is an alternating sequence of vertices and edges

$$u = v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k = v$$

such that $e_i = \{v_i, v_{i+1}\}$ for $i = 1, \ldots, k$

- A walk **starts** and **ends** with a **vertex**.
- A walk can travel over any edge and any vertex any number of times.
- If **no edge** appears more than once, the walk is a **path**.
- If **no vertex** appears more than once, the walk is a **simple path**.

Walking in circles

A **closed walk** in a graph **G = (V, E)** is a walk

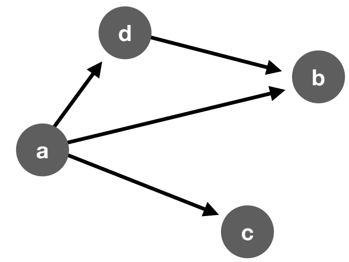
$$v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k$$

such that $v_0 = v_k$

- A **circuit** is a **path** where $v_0 = v_k$ (no repeated edges)
- A **cycle** is a **simple path** where $v_0 = v_k$ (no repeated vertices except v_0)
- The **length** of a walk is the number of edges in the sequence.

Walking on graphs vs digraphs

In a **directed graph**, a walk can only follow the **direction of the arrows**.



There is **no directed walk** from **b** to **a**.

Useful theorems

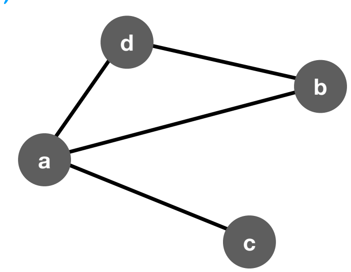
(about undirected graphs)

- If there is a **walk** from **u** to **v**, then there is a **walk** from **v** to **u**.
- If there is a **walk** from **u** to **v**, then there is a **path** from **u** to **v** (and from **v** to **u**).
- If there is a **path** from **u** to **v**, then there is a **simple path** from **u** to **v** (and **v** to **u**).
- Every **circuit** through **v** contains a **cycle** through **v**.
- Not every **closed walk** through **v** contains a **cycle** through **v**.

Degree

The **degree** of a vertex **v** is the **number of edges** incident to **v**.

Denoted: `deg(v)`

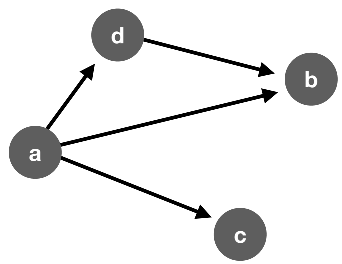


What is the degree of **c**? of **a**?

Degree on Digraphs

The **in-degree** of a vertex **v** is the **number of incoming edges** incident to **v**.

Denoted: `in-deg(v)`

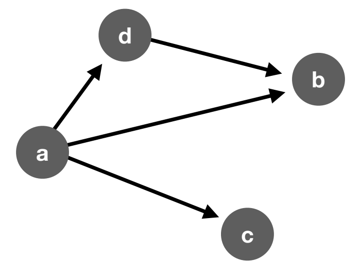


What is the in-degree of **c**? of **a**?

Degree on Digraphs

The **out-degree** of a vertex **v** is the **number of outgoing edges** incident to **v**.

Denoted: `out-deg(v)`



What is the out-degree of **c**? of **a**?

Degree theorem

For any graph **G = (V, E)**

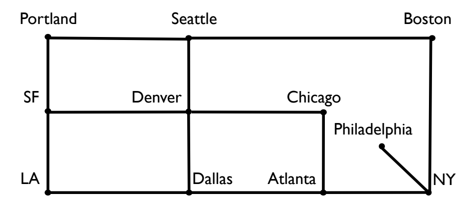
$$\sum_{v \in V} \deg(v) = 2\,|E|$$

where **|E|** is the number of edges in **G**.

Proof: by induction on **|E|**.

Hint: How does **removing an edge** change the equation?

Activity



Walk:
ex:

Path:
ex:

Simple path:
ex:

Closed Walk:
ex:

Circuit:
ex:

Cycle:
ex:

Degree:
Max Degree Vertex:
Min Degree Vertex:

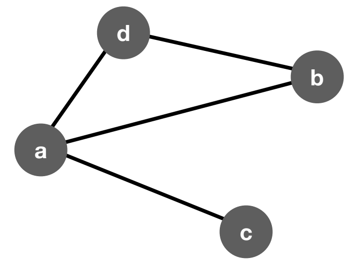
Reachability and Connectedness

"Siri, can I drive from Boston to Hong Kong?"

"Siri, can I drive from any point to any other point?"

Reachability

A vertex **v** in **G** is **reachable** from vertex **u** in **G** if there is a **path** from **u** to **v**.

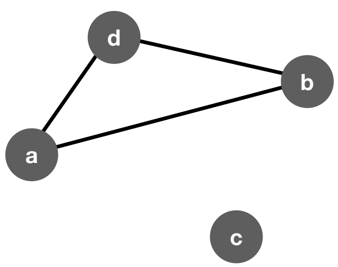


For an **undirected** graph **G**, **v** is **reachable** from vertex **u** iff **u** is **reachable** from vertex **v**.

Is **c** **reachable** from **d**? Yes.

Connectedness

An undirected graph **G** is **connected** if for every pair of vertices **u, v** in **G**, **v** is **reachable** from **u**.



The set of all **vertices reachable from v**, along with all **edges** of **G** connecting any two of them, is called the **connected component of v**.

(note that the connected component is itself a graph)

Recap & Next Class

Today:

Graphs

Next class:

Graph operations

Graph representations