CSE 373: Data Structures and Algorithms

Lecture 14: Introduction to Graphs

Instructor: Lilian de Greef
Quarter: Summer 2017

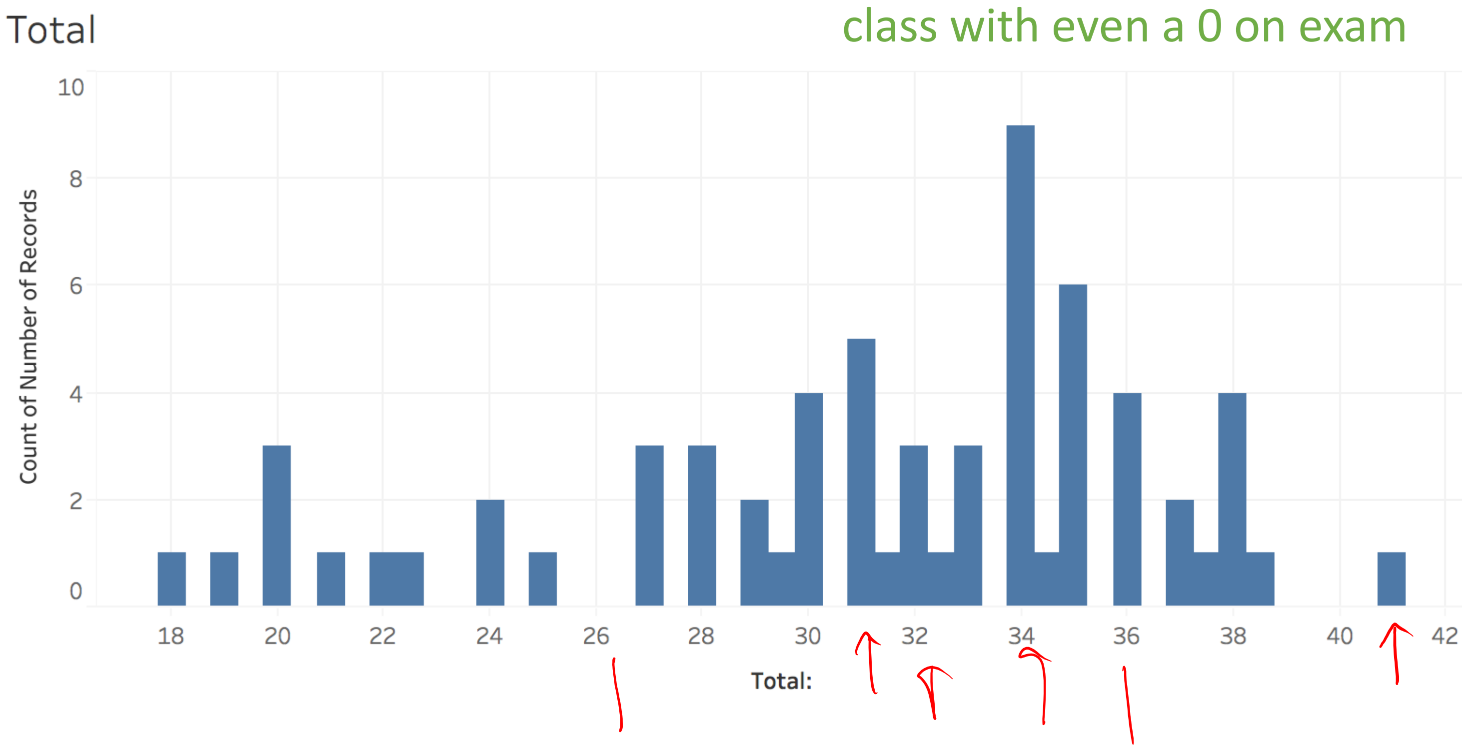
Today

- Overview of Midterm
- Introduce Graphs
  - Mathematical representation
  - Undirected & Directed Graphs
  - Self edges
  - Weights
  - Paths & Cycles
  - Connectedness
  - Trees as graphs
  - DAGs
  - Density & Sparsity

Midterm: Statistics and Distribution

Remember: it's curved

20% of grade → can pass class with even a 0 on exam



Mean 31.2 /43

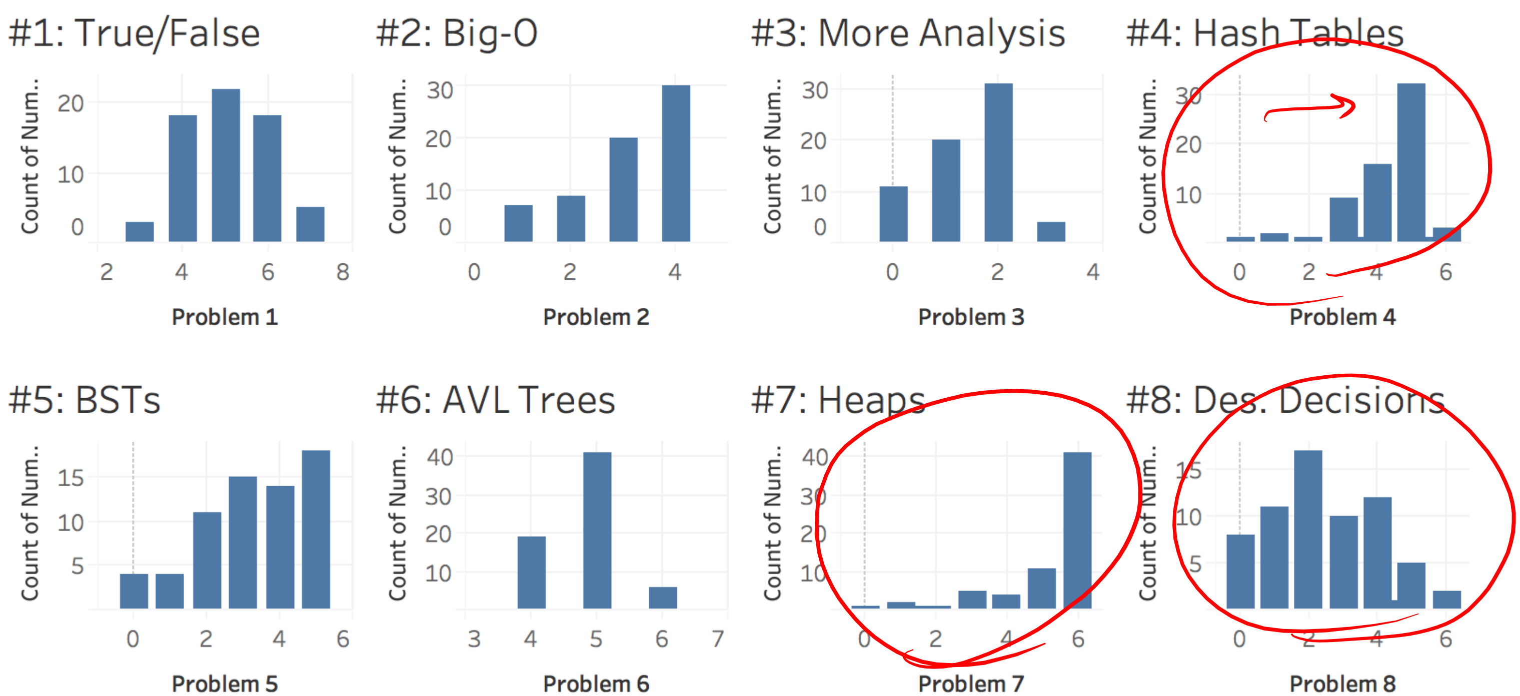
Std. dev. 5.48

Median 32.5 /43

Mode 34 /43

Max 41 /43

Midterm: Distribution by Problem



Hash Tables

There is a hash table implemented with linear probing that doubles in size every time its load factor is strictly greater than 1⁄2.

What is the worst-case condition for insert in this table?

- rehash (copy over n items)
- cluster of size n when rehash

What is the asymptotic worst-case running time to insert an item? *(let n = # items in table)*

$O(n^2)$

What is the amortized running time to insert an item to this table?

$\frac{n^2}{n} = O(n)$

Hash Tables

Now we have a hash table implemented with separate chaining in which each chain stores its keys in sorted order.

What is the worst-case condition for insert in this table?

- rehash
- insert $O(n)$ (sorting)

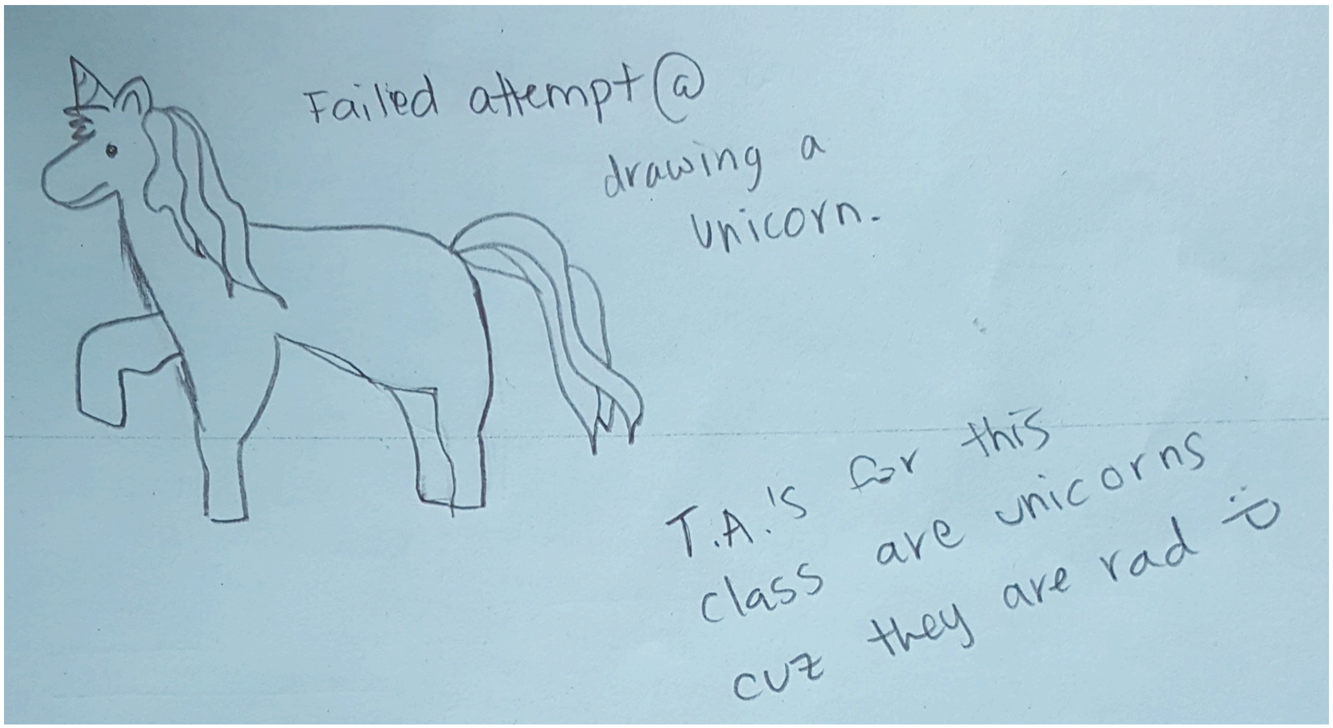
What is the asymptotic worst-case running time to insert an item into this table?

$n \times O(n) = O(n^2)$

Questions on Midterm?

- We’re happy to go over answers with you!
- Come visit any of our office hours ☺
- We encourage it! Mistakes are one of the best ways to learn.

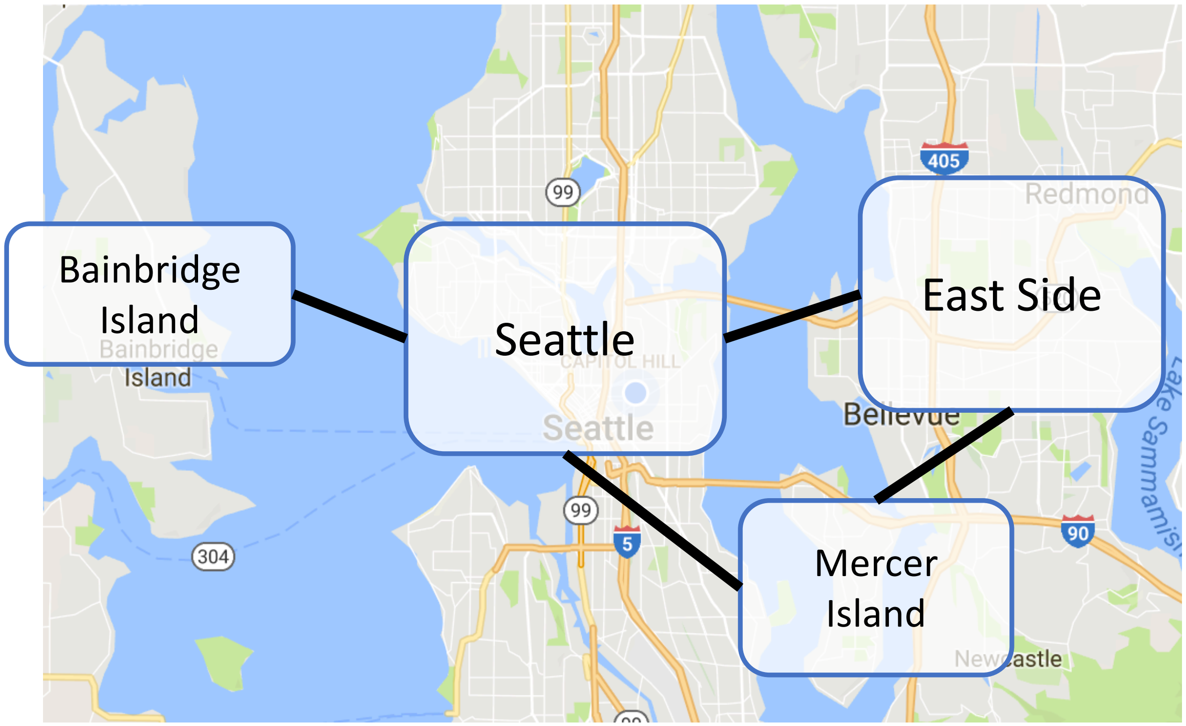
Fun drawing on last page of a midterm:



Introducing: Graphs

Vertices, edges, and paths (oh my!)

Introductory Example



This representation is called a graph

In this example, locations (Seattle, Bainbridge Island, the East Side, and Mercer Island) are the vertices (nodes)

And the roads, bridges, and ferry lines are the edges

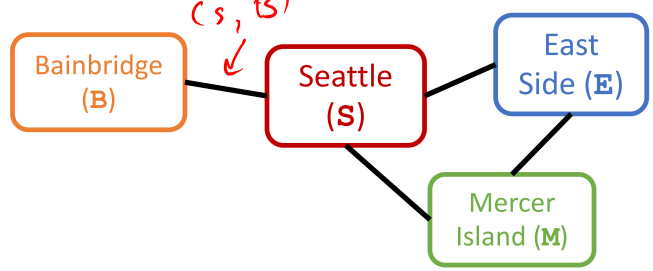
Graphs

Moon ☾

- A graph is a formalism for representing relationships among items
  - Very general definition because very general concept
- A **graph** is a pair $G = (V, E)$
  - A set of **vertices**, also known as nodes
    $V = \{v_1, v_2, \ldots, v_n\}$
  - A set of **edges**
    $E = \{e_1, e_2, \ldots, e_m\}$
    - An edge "connects" the vertices
    - Each edge e_i is a pair of vertices

$$e_i = (v_j, v_k)$$

(S, B)



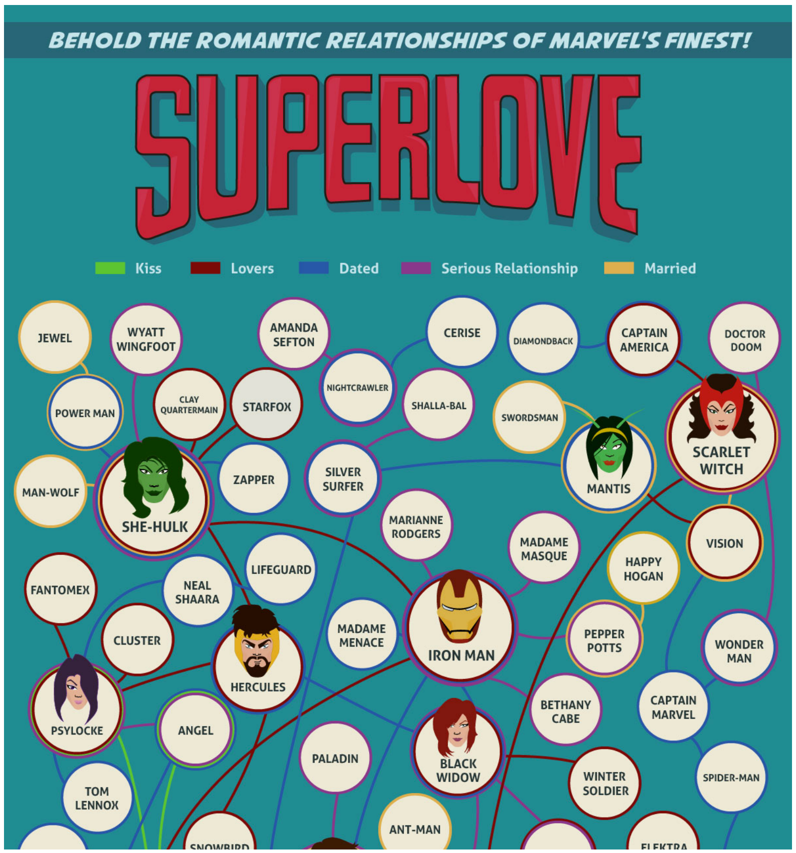
$V = \{S, M, E, B, ☾\}$

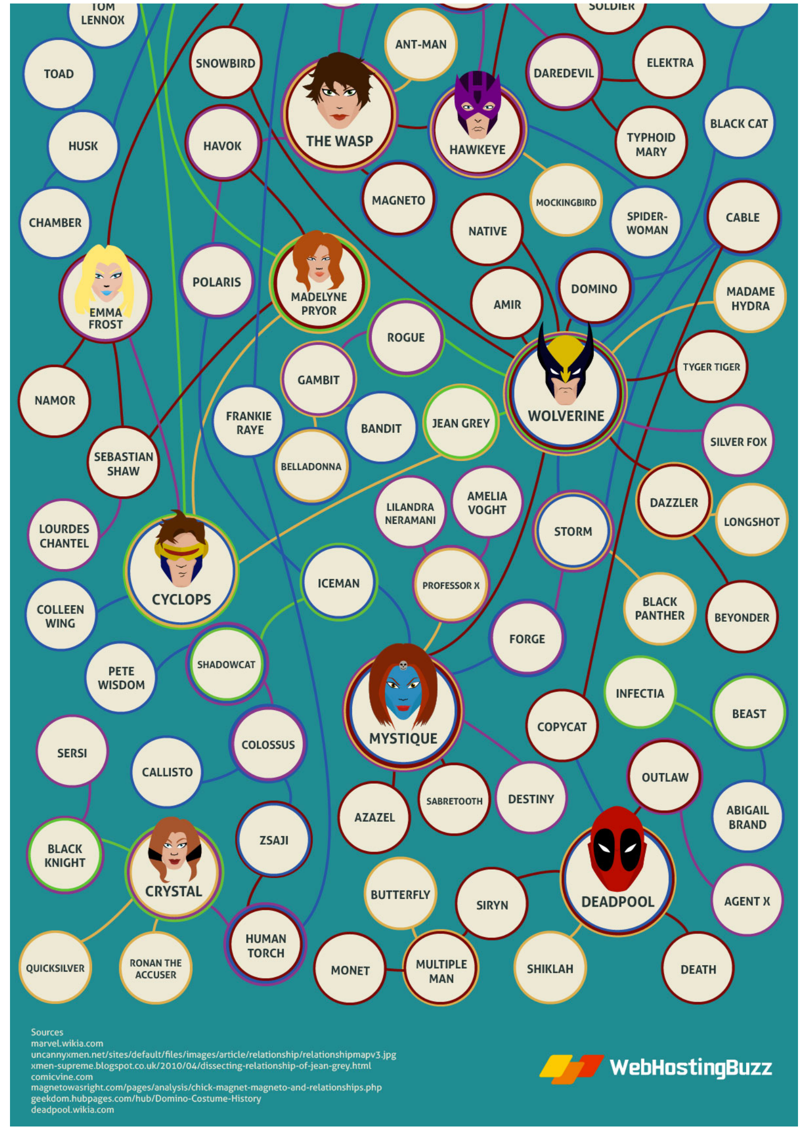
$E = \{ (S, B),$ = (B, S)?
$(S, E),$
$(S, M),$
$(M, E) \}$

- Graphs can be directed or undirected

Another Example:

($\mathbb{V}$ = { characters }, $\mathbb{E}$ = { romances })

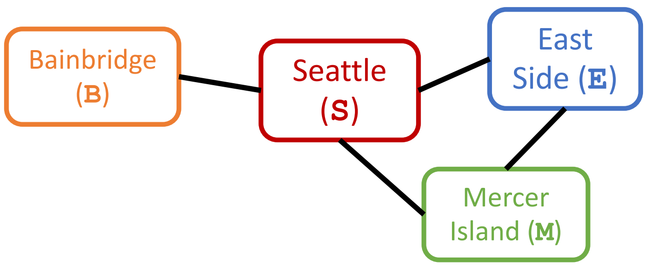


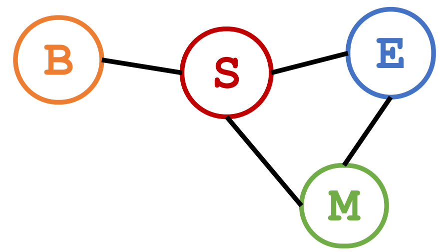


Source: http://www.webhostingbuzz.com/blog/2015/02/10/superlove-marvels-romantic-relationships-mapped/

Undirected Graphs

- In **undirected graphs**, edges have no specific direction
  - Edges are always "two-way"





- Thus, `(u,v)` $\in$ `E` implies `(v,u)` $\in$ `E`
  – Only one of these edges needs to be in the set
  – The other is implicit, so normalize how you check for it

$(M, E) \longleftrightarrow (E, M)$

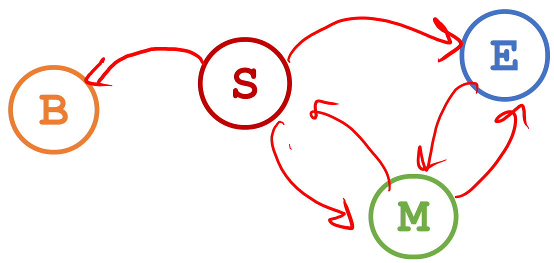
- **Degree** of a vertex: number of edges containing that vertex
  – Put another way: the number of adjacent vertices

degree(**S**) = 3

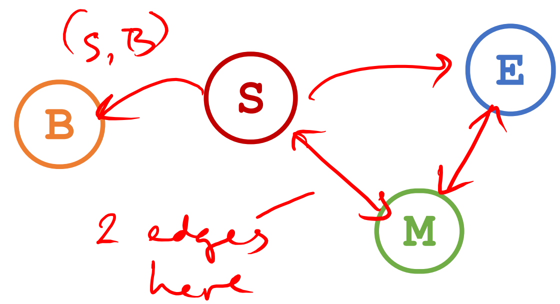
degree(**B**) = 1

Directed Graphs

- In **directed graphs** (sometimes called **digraphs**), edges have a direction



or



- Thus, `(u,v)` ∈ `E` does *not* imply `(v,u)` ∈ `E`.
  - Let `(u,v)` ∈ `E` mean `u` → `v`
  - Call `u` the **source** and `v` the **destination**

(S,B)
source
destination

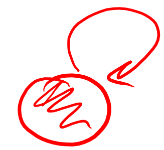
- **In-degree** of a vertex: number of in-bound edges, i.e., edges where the vertex is the destination

In-degree(**E**) = 2

- **Out-degree** of a vertex: number of out-bound edges i.e., edges where the vertex is the source

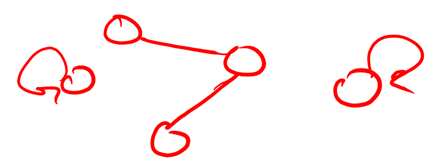
Out-degree(**B**) = 0

Self-Edges, Connectedness



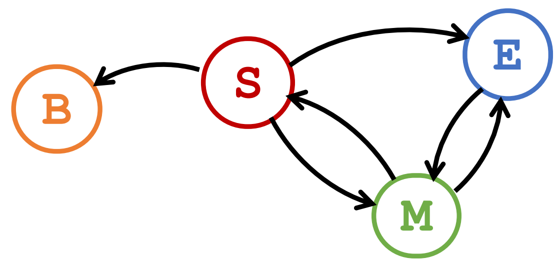
in-degree = 1
out-degree = 1

- A **self-edge** a.k.a. a **loop** is an edge of the form `(u,u)`
  - Depending on the use/algorithm, a graph may have:
    - No self edges
    - Some self edges
    - All self edges (often therefore implicit, but we will be explicit)
- A node can have a degree / in-degree / out-degree of zero
- A graph does not have to be connected
  - Even if every node has non-zero degree



More notation

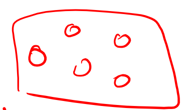
For a graph `G = (V,E)`:

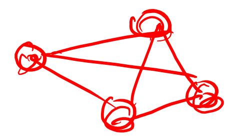


`V = {S,M,E,B}`
`E = {(S,B),`
`(S,E),`
`(S,M),`
`(M,E)}`

- `|V|` is the number of vertices
- `|E|` is the number of edges
  - Minimum? 0
  - Maximum for undirected? $|V||V-1|/2$
  - Maximum for directed? $|V|^2 \in O(|V|^2)$

(assuming self-edges allowed, else subtract `|V|`)





$(n-1)+(n-2)+(n-3)\dots$

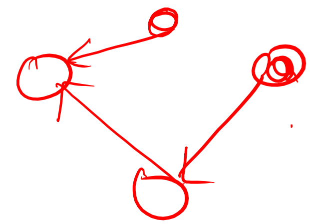
- If `(u,v) ∈ E`
  - Then `v` is a **neighbor** of `u`, i.e., `v` is **adjacent** to `u`
  - Order matters for directed edges
    - `u` is not **adjacent** to `v` unless `(v,u) ∈ E`

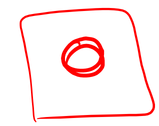
Is **M** adjacent to **B**? No

Is **S** adjacent to **B**? No

Is **B** adjacent to **S**? Yes!

Examples





Which would…

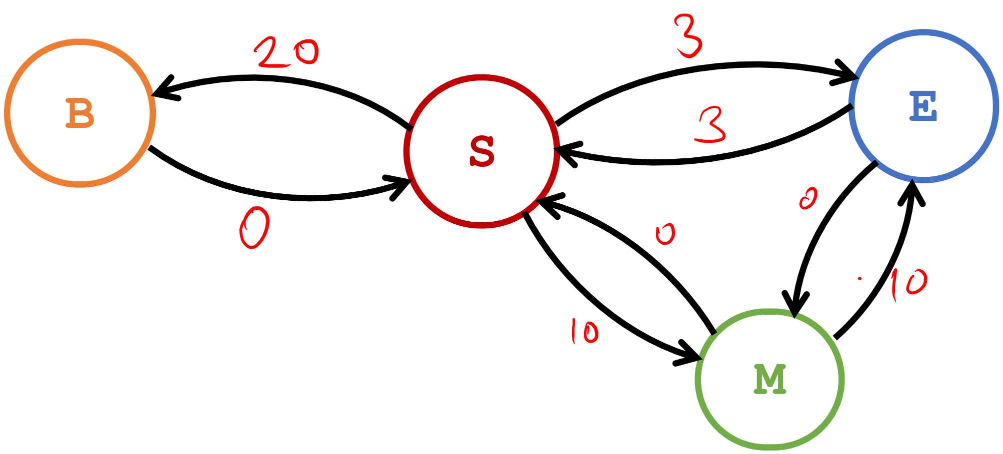
Use directed edges? Have self-edges? Be connected? Have 0-degree nodes?

(a) (b) (c) (d)

1. Web pages with links (a, b, c)
2. Facebook friends V= users E = friendship (c, d)
3. Methods in a program that call each other V= methods E= calls (a, b, c)
4. Road maps (e.g., Google maps) (a, b, c)
5. Airline routes (a, [b], c)
6. Family trees (a, c)
7. Course pre-requisites (a, c)

Weighted Graphs

- In a weighed graph, each edge has a **weight** a.k.a. **cost**
  - Typically numeric (most examples use ints)
  - Some graphs allow *negative weights*; many do not



Examples

What, if anything, might weights represent for each of these?
Do negative weights make sense?

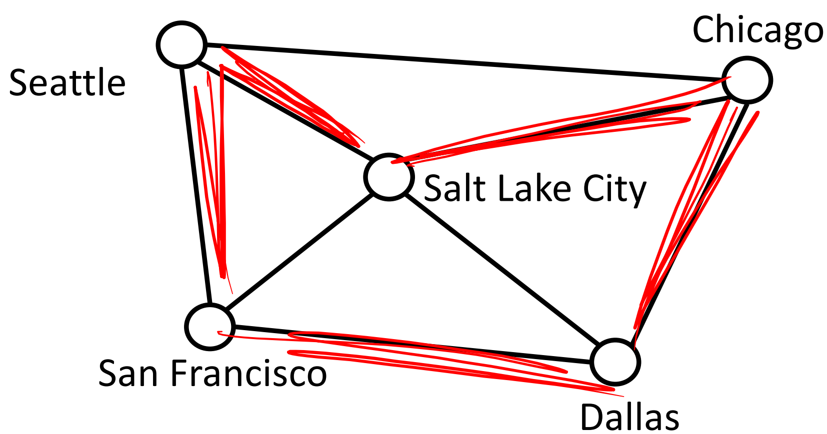
bookmarks vs not bookmarks vs visited. loading time

- Web pages with links ← bookmarks
- Facebook friends
- Methods in a program that call each other
- Road maps (e.g., Google maps) ← distance
- Airline routes ← time, cost, # passengers
- Family trees ← type of relation
- Course pre-requisites

Paths and Cycles

- A **path** is a list of vertices $[v_0, v_1, \ldots, v_n]$ such that $(v_i, v_{i+1}) \in E$ for all $0 \le i < n$. Said as *"a path from* v_0 *to* v_n*"*

- A **cycle** is a path that begins and ends at the same node ($v_0 == v_n$)



Example: [Seattle, Salt Lake City, Chicago, Dallas, San Francisco, Seattle]

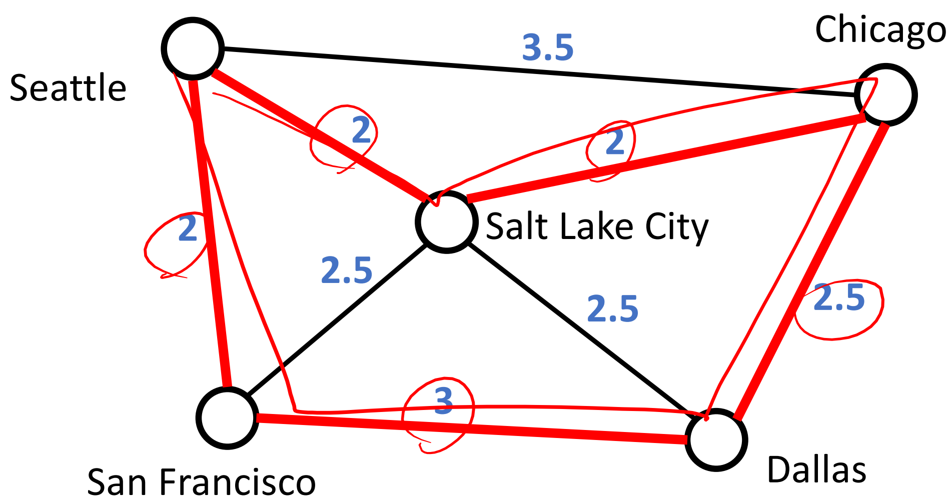
Path Length and Cost

Path length: Number of *edges* in a path

Path cost: Sum of *weights* of edges in a path

Example:

let **P** = [Seattle, Salt Lake City, Chicago, Dallas, San Francisco, Seattle]



length(**P**) = 5

cost(**P**) = 11.5

Simple Paths and Cycles

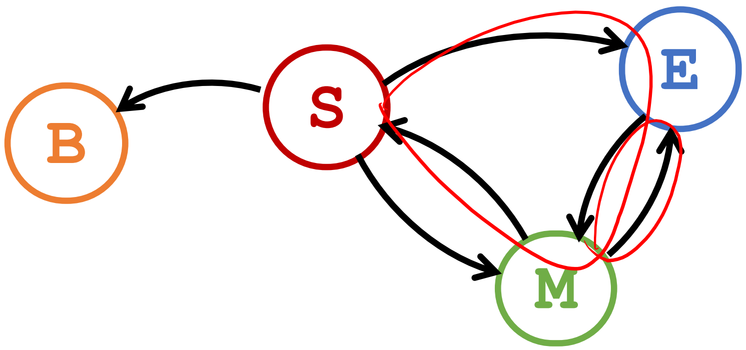
- A **simple path** repeats no vertices, except the first might be the last
  e.g. [Seattle, Salt Lake City, San Francisco, Dallas]
  [Seattle, Salt Lake City, San Francisco, Dallas, Seattle]

- Recall, a **cycle** is a path that ends where it begins
  e.g. [Seattle, Salt Lake City, San Francisco, Dallas, Seattle]
  [Seattle, Salt Lake City, Seattle, Dallas, Seattle]

- A **simple cycle** is a cycle and a simple path
  e.g. [Seattle, Salt Lake City, San Francisco, Dallas, Seattle]

Paths and Cycles in Directed Graphs

Example:

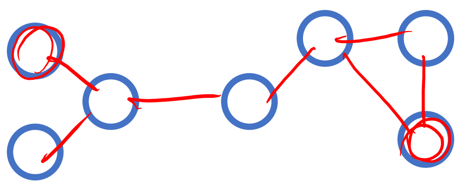


Is there a path from **B** to **M** ? No

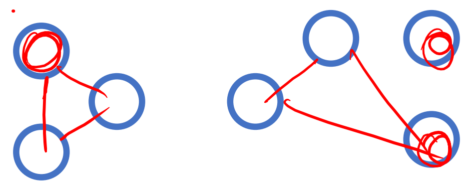
Does the graph contain any cycles? Yes

Undirected-Graph Connectivity

- An undirected graph is **connected** if
  for all pairs of vertices `(u,v)`, there exists a *path* from `u` to `v`

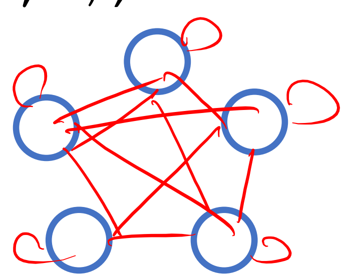


Connected graph



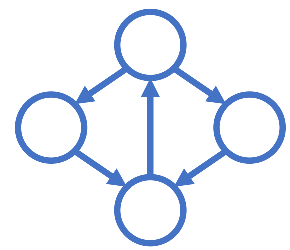
Connected graph

- An undirected graph is **complete**, a.k.a. **fully connected** if
  for *all* pairs of vertices `(u,v)`, there exists an *edge* from `u` to `v`

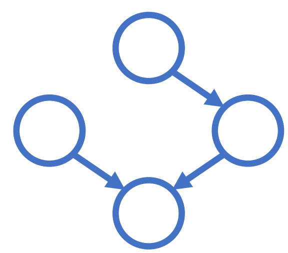


Directed-Graph Connectivity

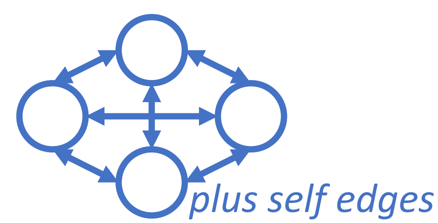
- A directed graph is **strongly connected** if there is a path from every vertex to every other vertex



- A directed graph is **weakly connected** if there is a path from every vertex to every other vertex *ignoring direction of edges*



- A **complete** a.k.a. **fully connected** directed graph has an edge from every vertex to every other vertex



Practice Time!

directed

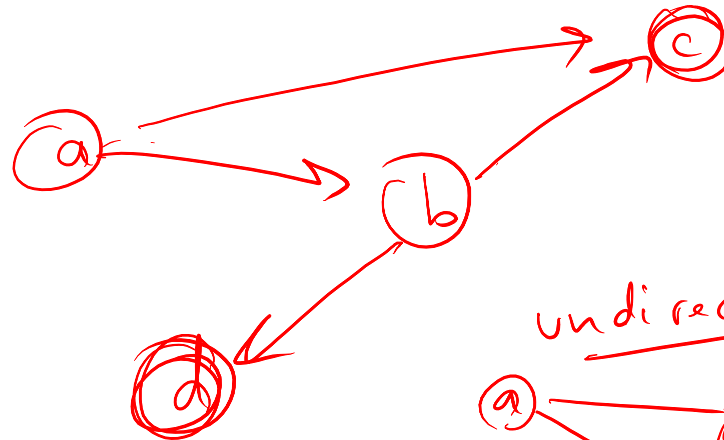
Let graph `G = (V, E)`

where

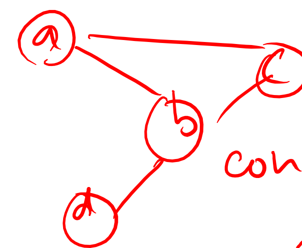
`V = {a, b, c, d}`

`E = {(a,b), (b,c), (a,c), (b,d)}`

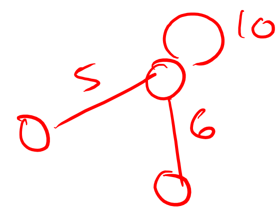
How connected is `G`?



undirected



connected



A. Disconnected

B. Weakly Connected

C. Strongly Connected

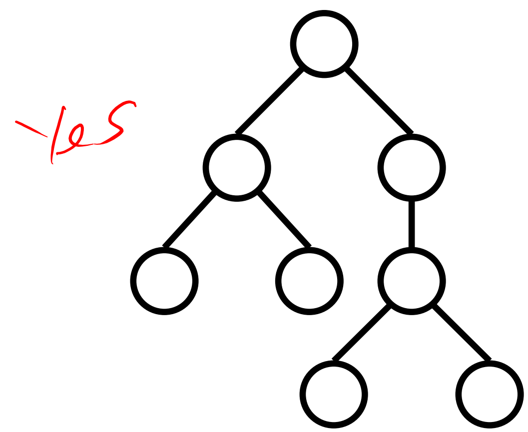
D. Complete / Fully Connected

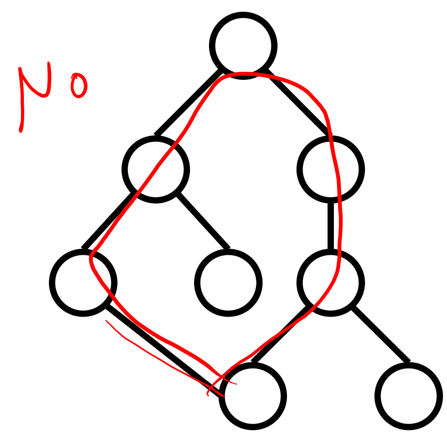
Trees as Graphs

When talking about graphs,

we say a **tree** is a graph that is:

- Connected
- Acyclic
  when you treat edges as undirected



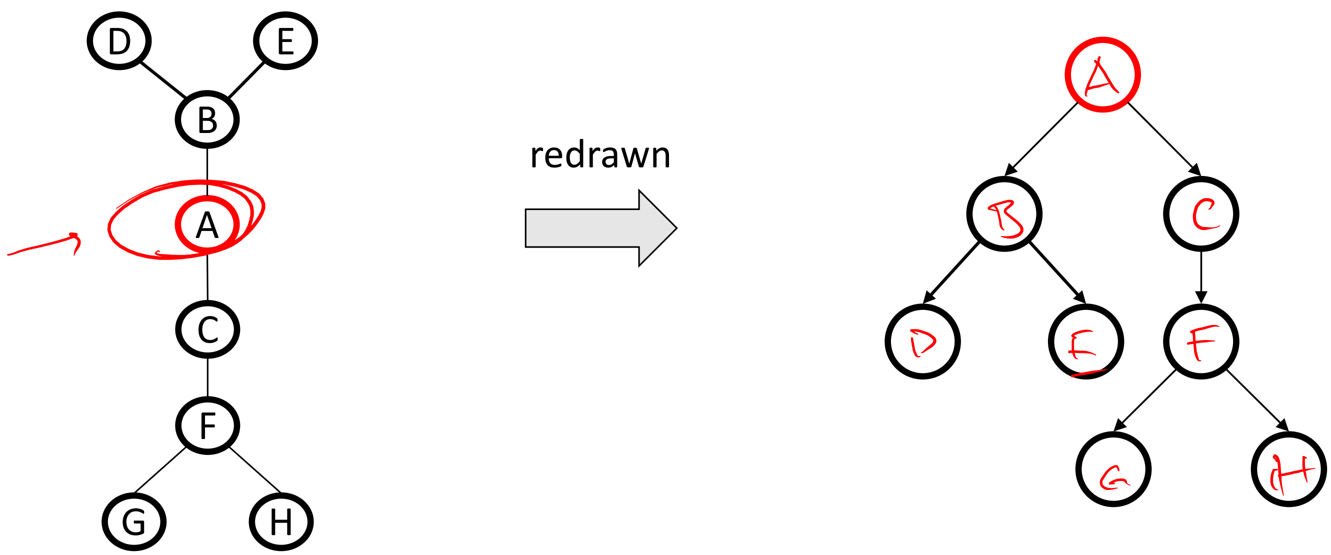


Note that

- Edges can be undirected (new!)
- All trees are graphs, but not all graphs are trees

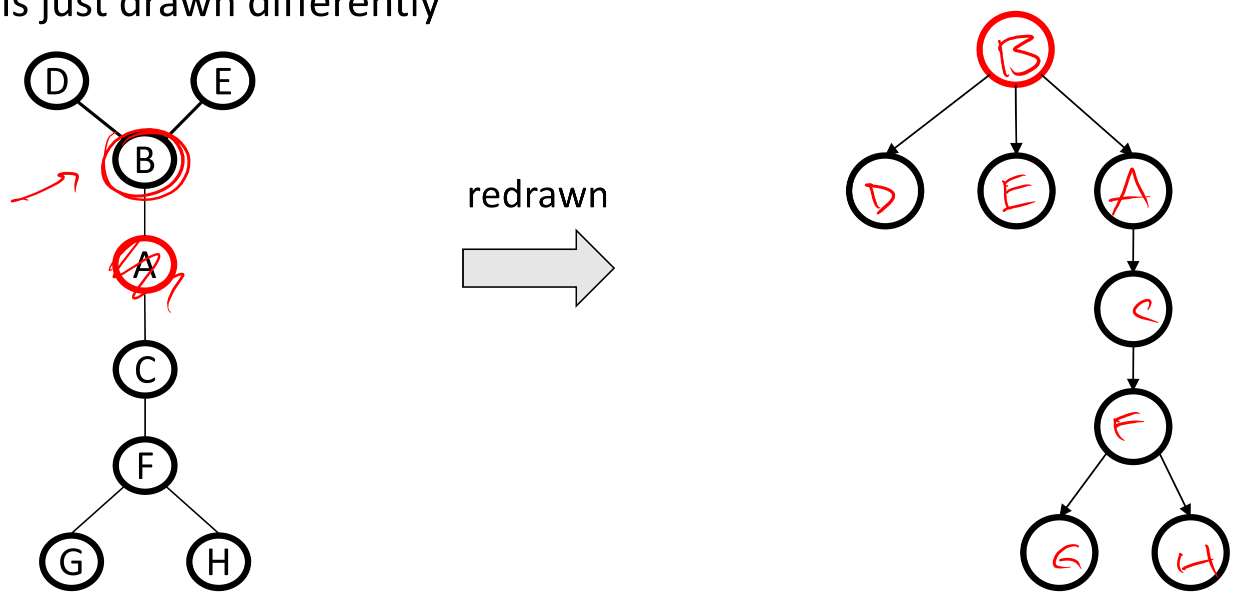
Rooted Trees

- We are more accustomed to **rooted trees** where:
  - We identify a unique root
  - We think of edges as directed: parent to children
- Given a tree, picking a root gives a unique rooted tree
  - The tree is just drawn differently



Rooted Trees

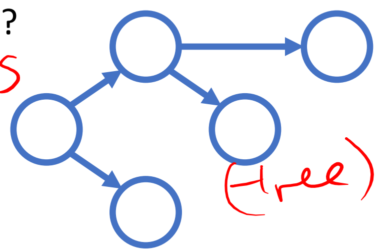
- We are more accustomed to **rooted trees** where:
  - We identify a unique root
  - We think of edges as directed: parent to children
- Given a tree, picking a root gives a unique rooted tree
  - The tree is just drawn differently



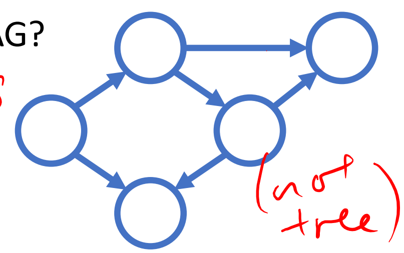
Directed Acyclic Graphs (DAGs)

- A **DAG** is a directed graph with no (directed) cycles
  - Every rooted directed tree is a DAG
  - But not every DAG is a rooted directed tree

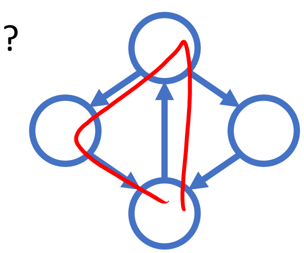
DAG?
Yes



DAG?
Yes



DAG?
No

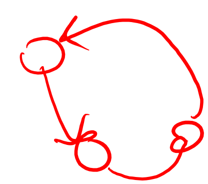


  - Every DAG is a directed graph
  - But not every directed graph is a DAG

Examples

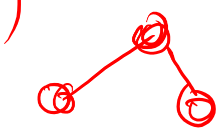
Which of our directed-graph examples do you expect to be a DAG?

- Web pages with links
- Methods in a program that call each other
- Airline routes
- Family trees
- Course pre-requisites

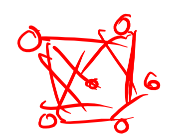


Density / Sparsity

- Recall: In an undirected graph, $0 \leq |E| < |V|^2$
- Recall: In a directed graph: $0 \leq |E| \leq |V|^2$
- So for any graph, $O(|E|+|V|)$ is $O(|E|) + O(|V|) = O(|V|^2)$ $|V|^2$



- Another fact: If an undirected graph is *connected*, then $|V|-1 \leq |E|$
- Because $|E|$ is often much smaller than its maximum size, we do not always approximate $|E|$ as $O(|V|^2)$
  - This is a correct bound, it just is often not tight
  - If it is tight, i.e., $|E|$ is $\theta(|V|^2)$ we say the graph is **dense**
    - More sloppily, dense means "lots of edges"



  - If $|E|$ is $O(|V|)$ we say the graph is **sparse**
    - More sloppily, sparse means "most possible edges missing"