CSE 332: Data Structures and Parallelism

Fall 2022

~~Richard Anderson~~

Anjali Agarwal

Lecture 17: Intro to Graph Theory

Announcements

- Upcoming lectures
  - Intro to graphs
  - Topological Sort
  - Parallelism (3 lectures)
  - Concurrency (2 lectures)
- Shift in lecture order to provide background for Project 3

Graphs

A formalism for representing relationships between objects

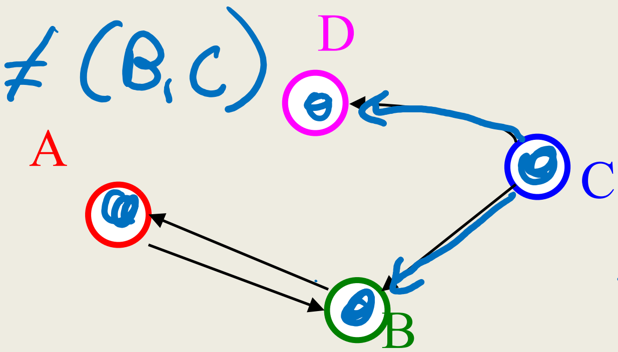
– Graph **G = (V,E)**

– *Set of vertices*:
V = {v_1, v_2, …, v_n}

– *Set of edges*:
E = {e_1, e_2, …, e_m}
where each **e_i** connects one

– vertex to another **(v_j, v_k)**

(C, B) ≠ (B, C)



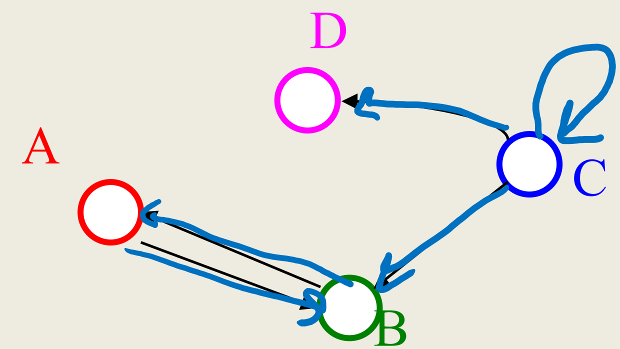
V = {A, B, C, D}
E = {(C, B),
(A, B),
(B, A)
(C, D)}

For *directed edges*, **(v_j, v_k)** and **(v_k, v_j)** are distinct.

|S|

Graphs

(c,c)



Notation

– **|V|** = number of vertices 4

– **|E|** = number of edges 4

"B is adjacent to C"

~~"C is adjacent to B"~~

v is *adjacent* to **u** if **(u,v)** ∈ **E**

–*neighbor* of = adjacent to

–Order matters for directed edges

It is possible to have an edge **(v,v)**,

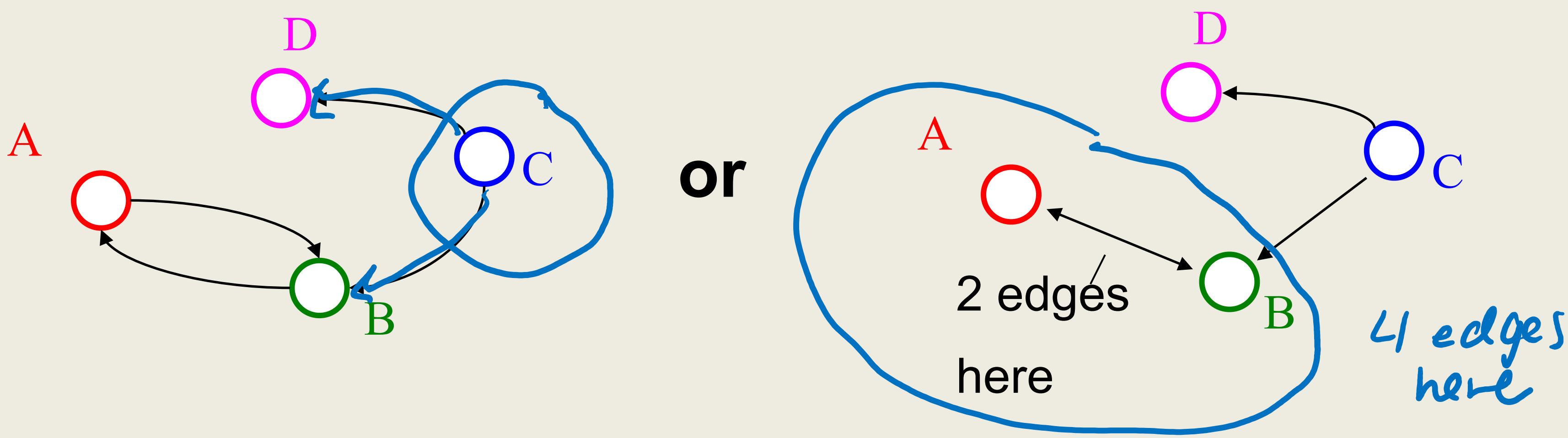
called a *loop*. "self-loop"

–We will assume graphs without loops.

```
V = {A, B, C, D}
E = {(C, B),
     (A, B),
     (B, A),
     (C, D)}
```

Directed Graphs

In *directed* graphs (a.k.a., *digraphs*), edges have a direction:

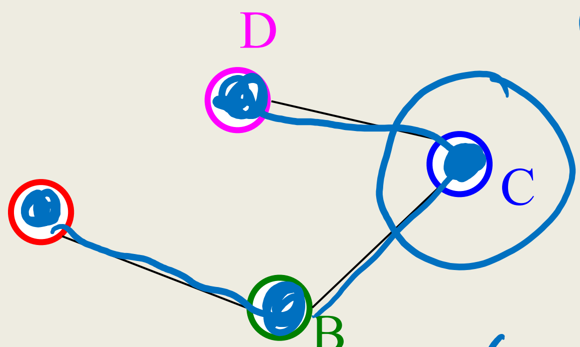


Thus, **(u,v)** ∈ **E** does *not* imply **(v,u)** ∈ **E**.
I.e., **v** adjacent to **u** does *not* imply **u** adjacent to **v**.

In-degree of a vertex: number of inbound edges. C : 0
Out-degree of a vertex : number of outbound edges. C : 2

Undirected Graphs

In *undirected* graphs, edges have no specific direction (edges are always two-way):



E = { (A,B), (B,C), (C,D) }

3 edges

(A,B) = (B,A)

Thus, **(u,v)** ∈ **E** *does* imply **(v,u)** ∈ **E**. Only one of these edges needs to be in the set; the other is implicit.

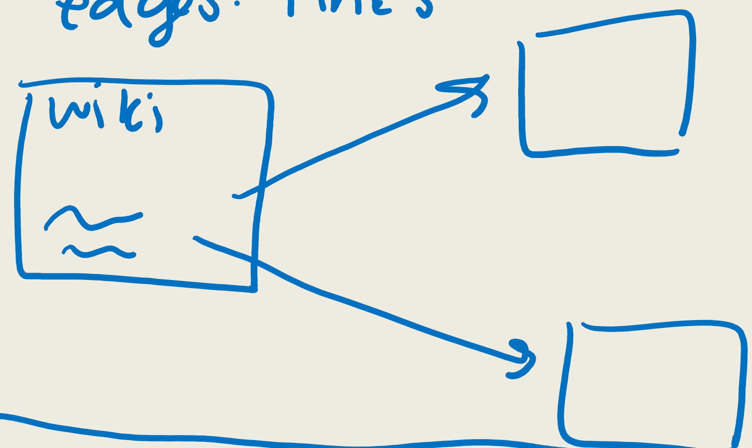
Degree of a vertex: number of edges containing that vertex. (Same as number of adjacent vertices.) degree c: 2

Examples of Graphs

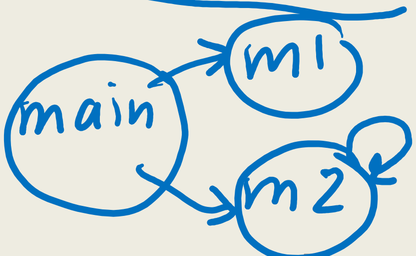
- For each, what are the vertices and edges? Are they directed or undirected?

- The internet
- Facebook ✓
- Highway map
- Airline routes
- Call graph of a program
- …

vertices: webpages
edges: links

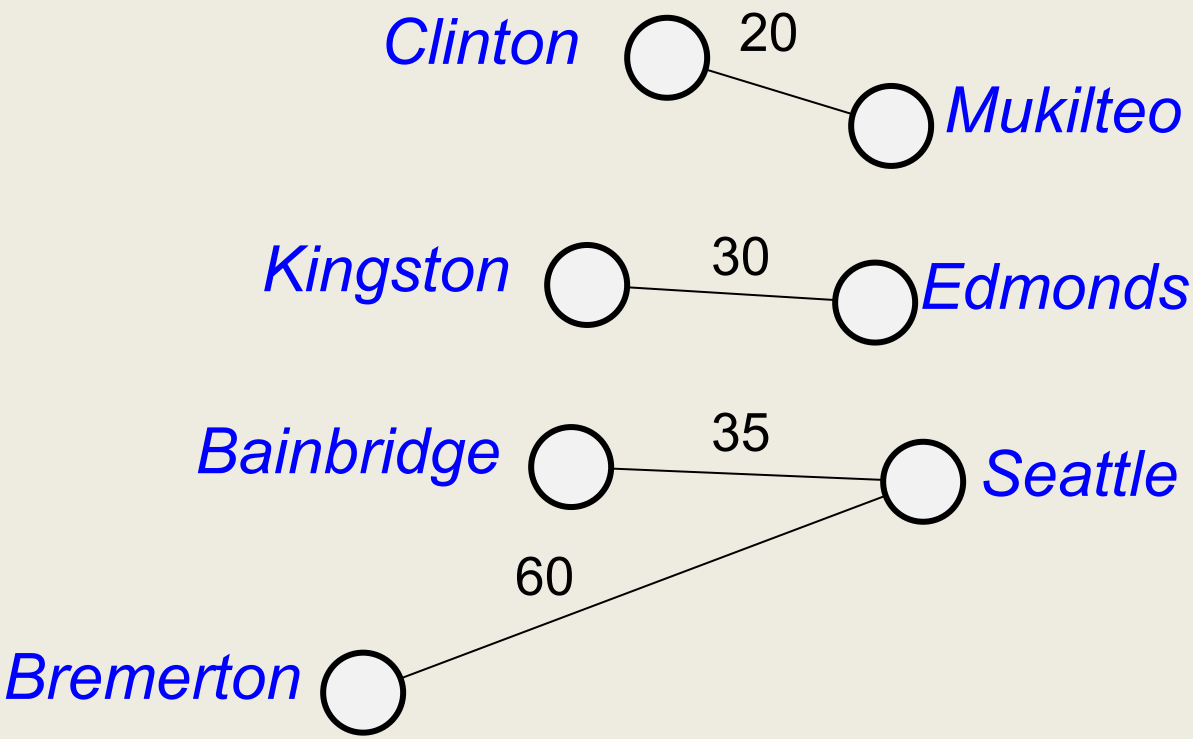


main()
m1()
m2()



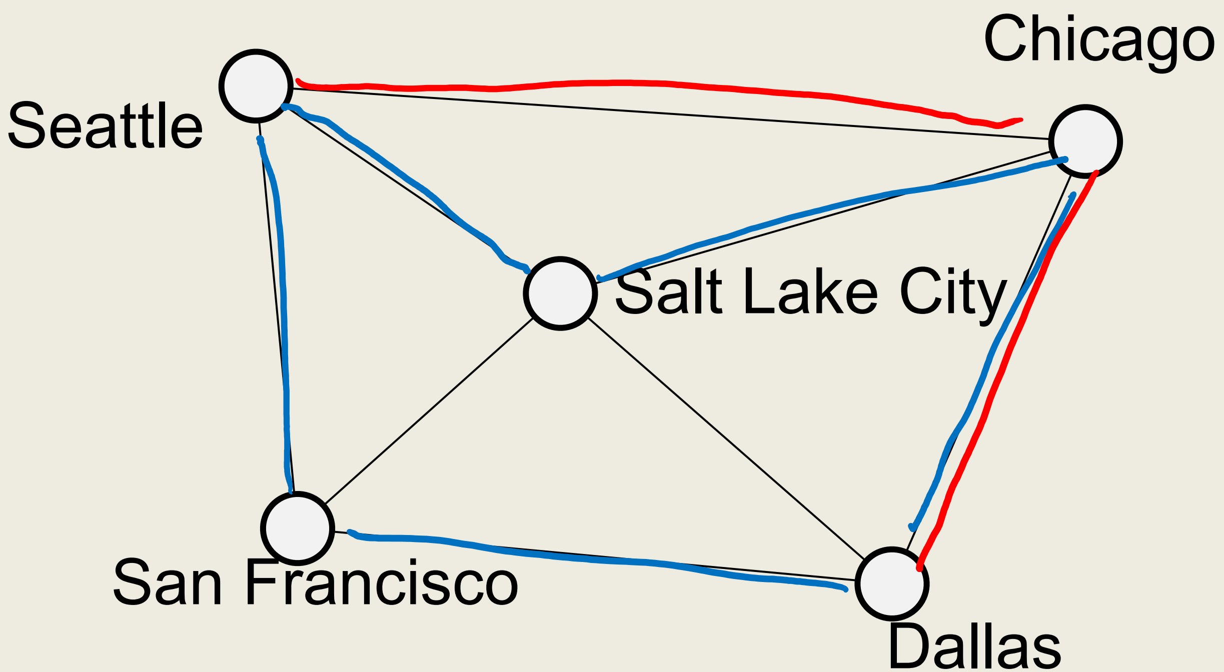
Weighted Graphs

Each edge has an associated weight or cost.



Paths and Cycles

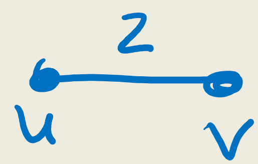
- A *path* is a list of vertices $\{w_1, w_2, \ldots, w_q\}$ such that $(w_i, w_{i+1}) \in E$ for all $1 \leq i < q$
- A *cycle* is a path that begins and ends at the same node

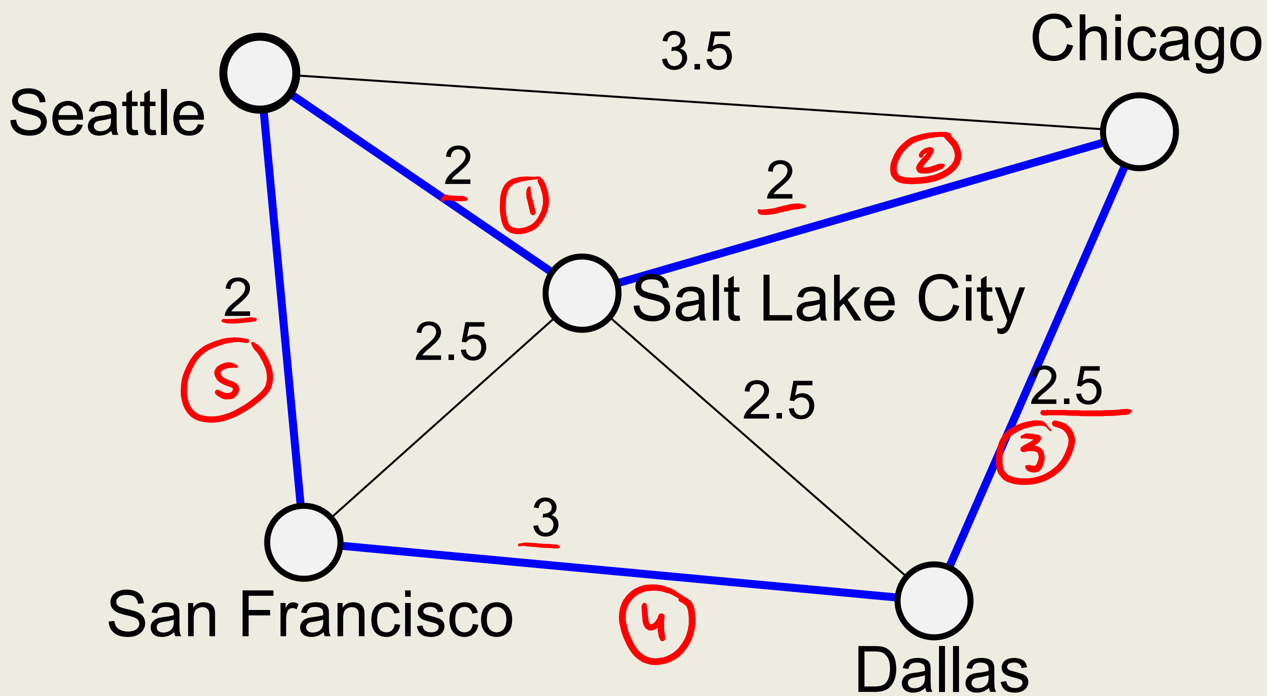


P = {Seattle, Salt Lake City, Chicago, Dallas, San Francisco, Seattle}

Path Length and Cost

- *Path length*: the number of edges in the path
- *Path cost*: the sum of the costs of each edge





For path P:

length(P) = 5

cost(P) = 11.5

✓ every edge has weight 1

How would you ensure that length(p)=cost(p) for all p?

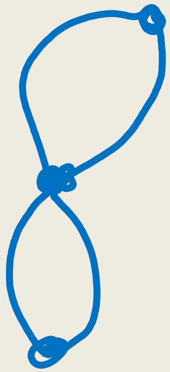
Simple Paths and Cycles

A *simple path* repeats no vertices (except that the first can also be the last):

- –P = {Seattle, Salt Lake City, San Francisco, Dallas}
- –P = {Seattle, Salt Lake City, Dallas, San Francisco, Seattle}

A *cycle* is a path that starts and ends at the same node:

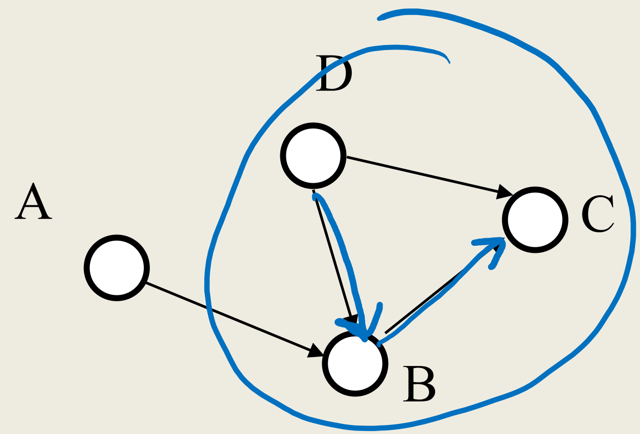
- –P = {Seattle, Salt Lake City, Dallas, San Francisco, Seattle}
- –P = {Seattle, Salt Lake City, Seattle, San Francisco, Seattle}



A *simple cycle* is a cycle that is also a simple path (in undirected graphs, no edge can be repeated).

Paths/Cycles in Directed Graphs

Consider this directed graph:

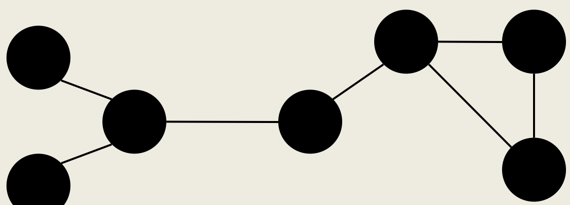


Is there a path from A to D? no

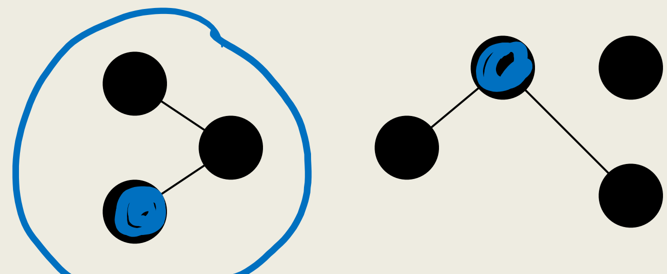
Does the graph contain any cycles? no

Undirected Graph Connectivity

- Undirected graphs are *connected* if there is a path between any two vertices:



Connected graph

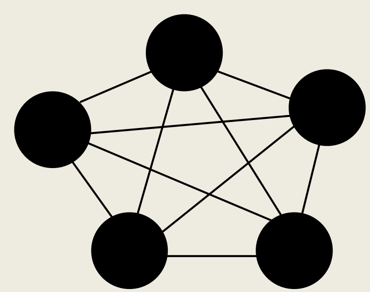


Disconnected graph

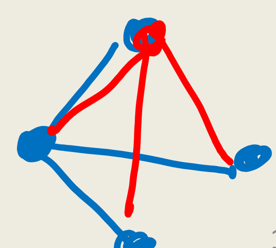
- A *complete undirected* graph has an edge between every pair of vertices:

n vertices

$$\frac{n(n-1)}{2}$$



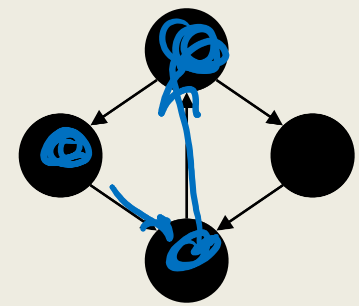
$\cancel{n!}$



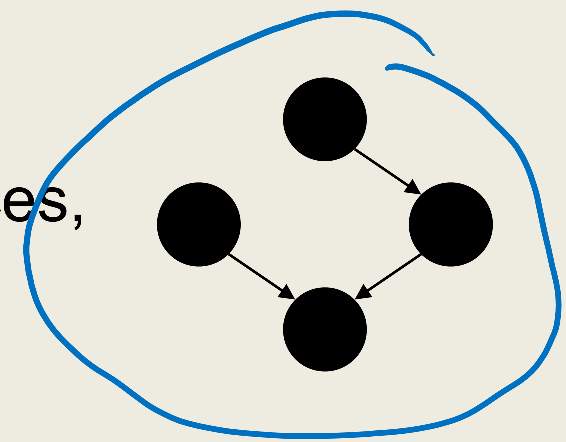
- (Complete = *fully connected*)

Directed Graph Connectivity

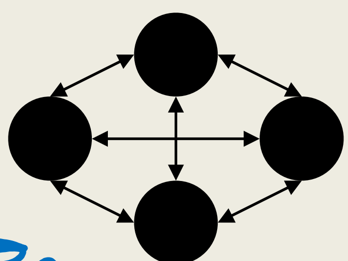
Directed graphs are *strongly connected* if there is a path from any one vertex to any other.



Directed graphs are *weakly connected* if there is a path between any two vertices, *ignoring direction.*



A *complete directed* graph has a directed edge between every pair of vertices. (Again, complete = *fully connected*.)

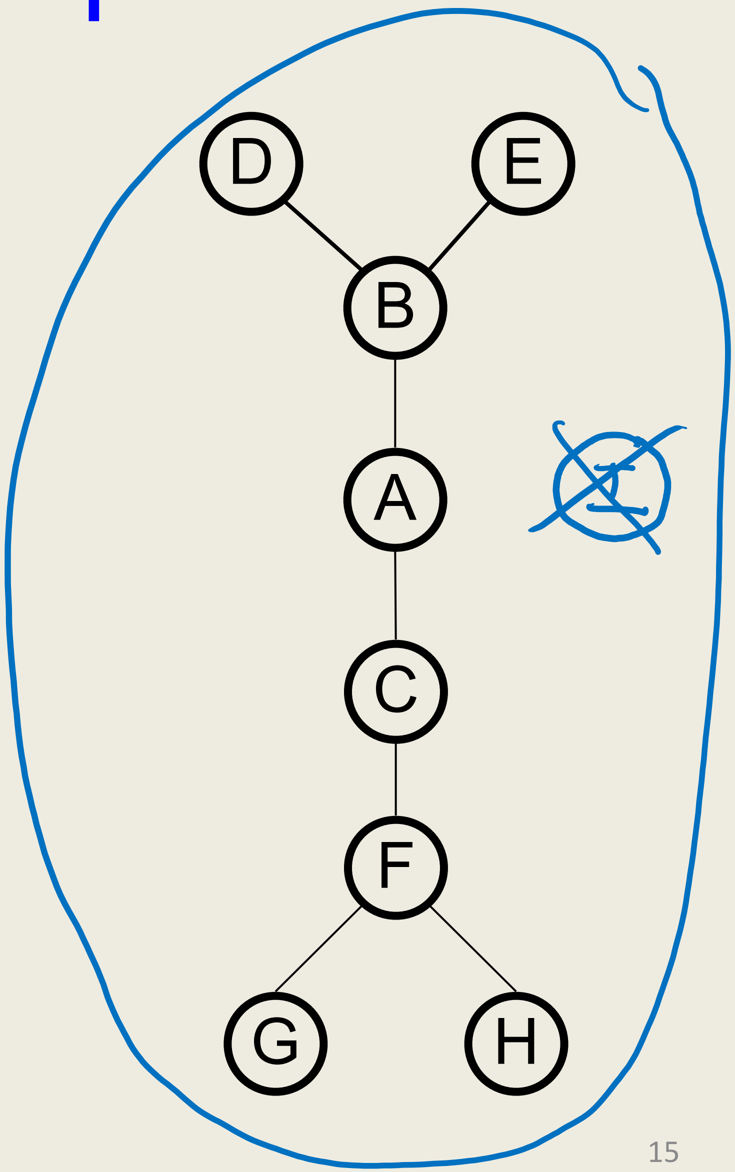


n vertices $\quad n(n-1)$

Trees as Graphs

A tree is a graph that is:

- *undirected*
- *acyclic*
- *connected*



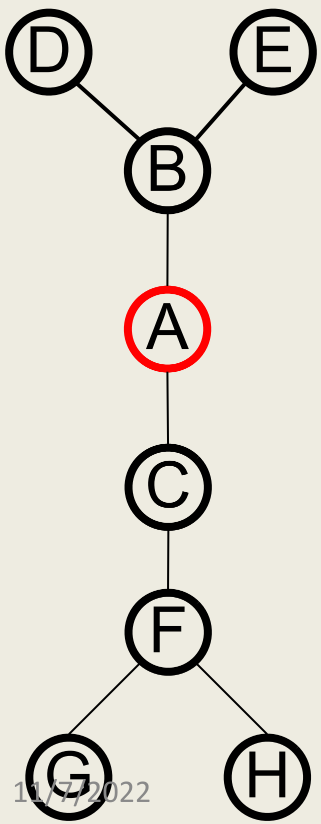
Rooted Trees

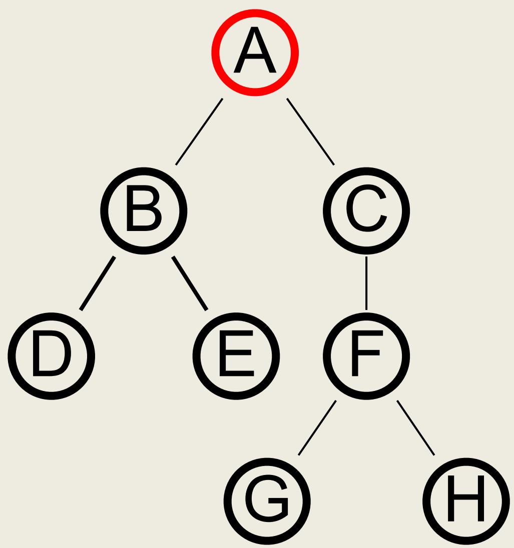
We are more accustomed to:

Rooted trees (a tree node that is "special")

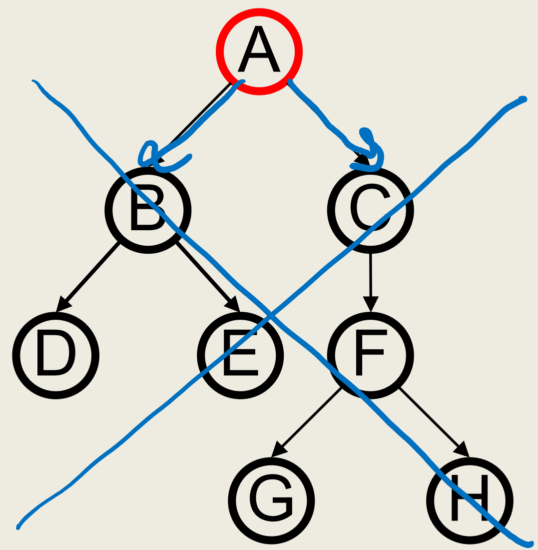
Directed edges from parents to children (parent closer to root).

A rooted tree (root indicated in red) drawn two ways



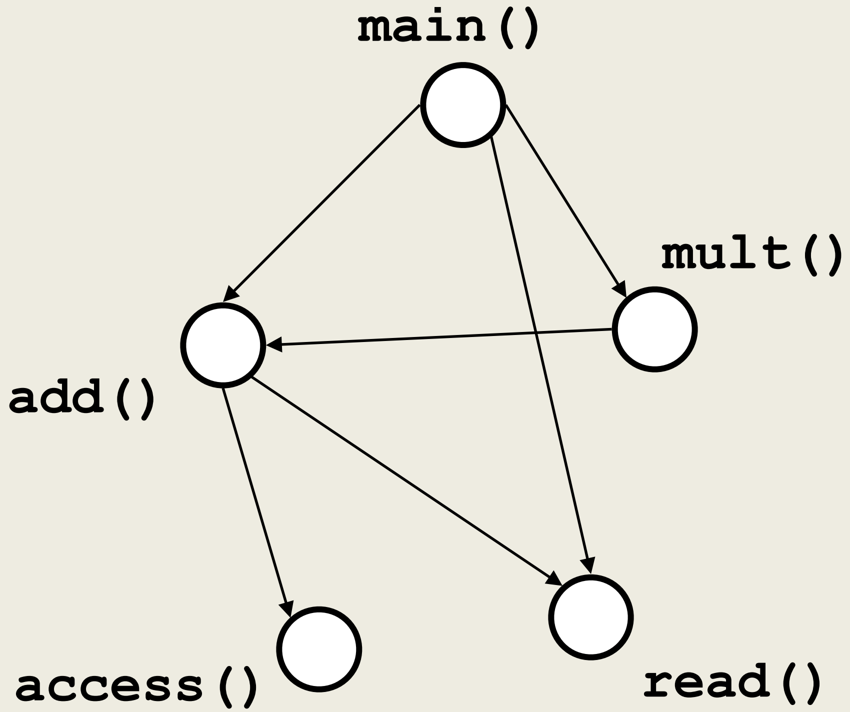


Rooted tree with directed edges from parents to children.



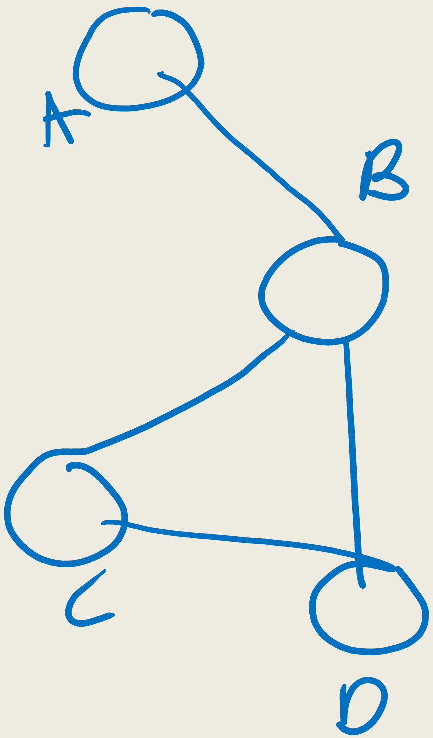
Directed Acyclic Graphs (DAGs)

- DAGs are directed graphs with no (directed) cycles.



What's the data structure?

Common query: which vertices are neighbors of a vertex



$$\cancel{V = \{A, B, C, D\}}$$

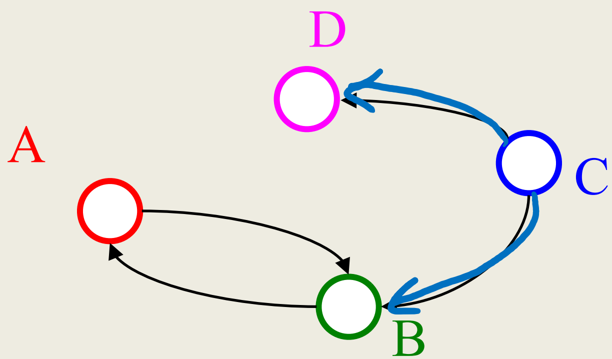
$$\cancel{E = \{(A,B), (B,C), (C,D), (B,D)\}}$$

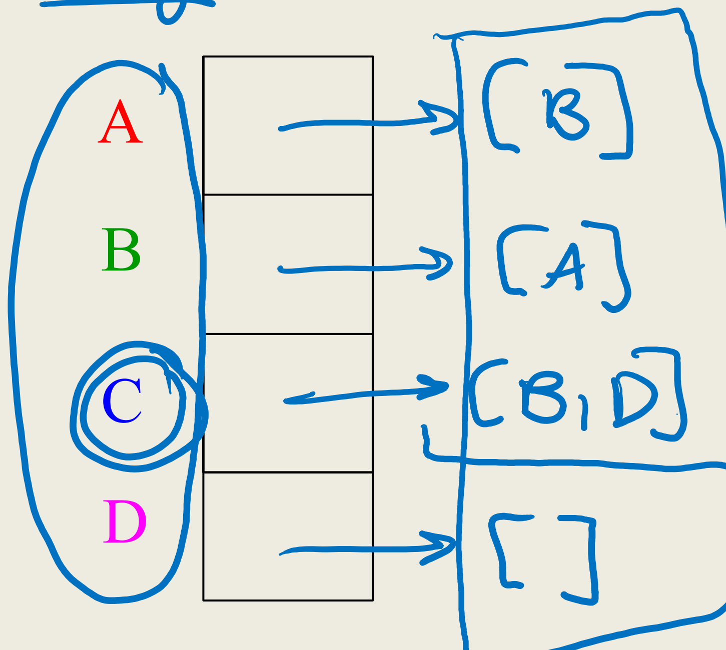
|V| : # vertices |E| : # edges d_v : out-degree of v

Representation 1: Adjacency List

A list (array) of length |V| in which each entry stores a list (linked list) of all adjacent vertices

array (hashmap)





Runtimes:

Iterate over vertices? $O(|V|)$

Iterate over edges? $O(|V| + |E|)$

Iterate edges adj. to vertex v? $O(d_v)$

Existence of edge (u,v)? $O(d_u)$

Space requirements? $O(|V| + |E|)$

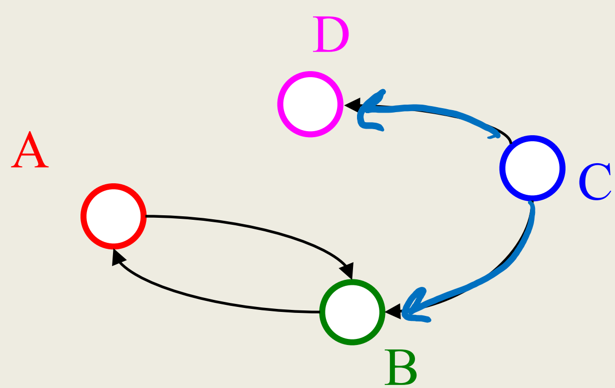
Best for what kinds of graphs? SPARSE

$|E| \leq |V| \cdot (|V|-1) < |V|^2$

Representation 2: Adjacency Matrix

A `|V| x |V|` matrix **M** in which an element **M[u,v]** is true if and only if there is an edge from **u** to **v**

True/False 1/0



	A	B	C	D
A	0	1	0	0
B	1	0	0	0
C	0	1	0	1
D	0	0	0	0

Runtimes:

Iterate over vertices? $O(|V|)$

Iterate over edges? $O(|V|^2)$

Iterate edges adj. to vertex? $O(|V|)$

Existence of edge? $O(1)$

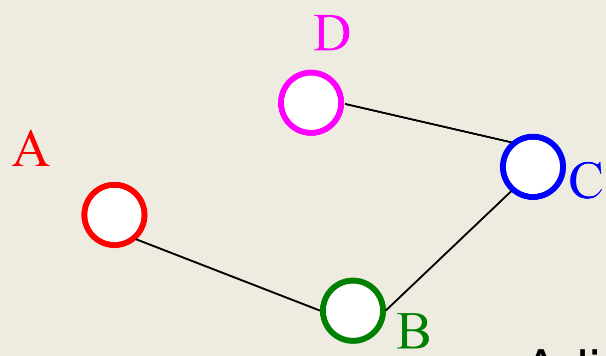
Space requirements? $O(|V|^2)$

Best for what kinds of graphs?

DENSE

Representing Undirected Graphs

What do these reps look like for an undirected graph?



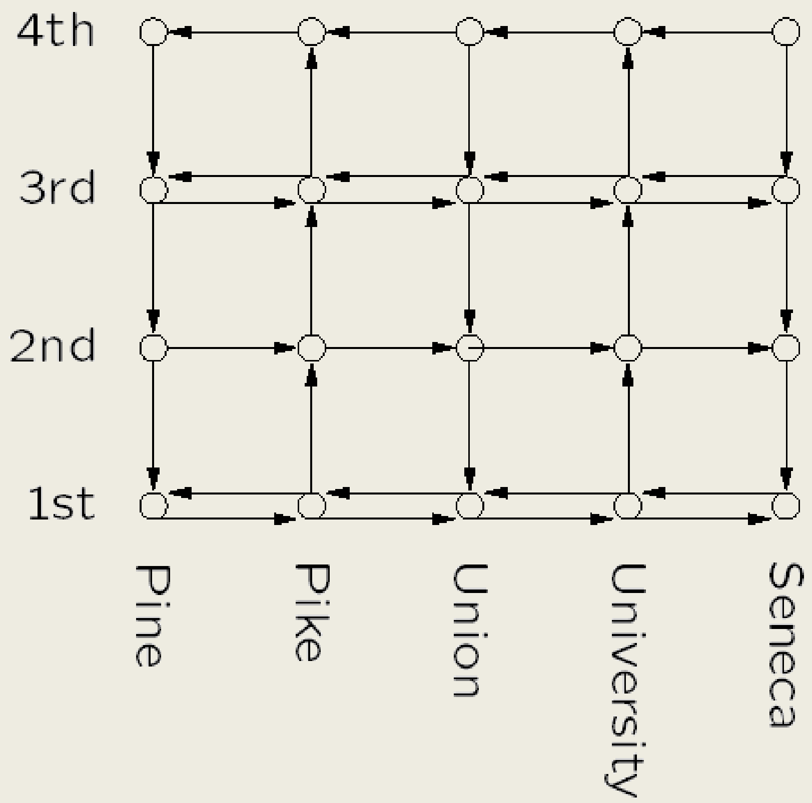
Adjacency matrix:

	A	B	C	D
A				
B				
C				
D				

Adjacency list:

A	
B	
C	
D	

Some Applications: Bus Routes in Downtown Seattle



If we're at 3rd and Pine, how can we get to 1st and University using Metro?

How about 4th and Seneca?