Algorithms and Data Structures for Data Science
Graph Implementations 3

CS 277
Brad Solomon

April 17, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Exam 3 Signups Available

April 24 — April 27

Very limited window for makeup exams (since end of semester is near)

Covers content from week 10 — 14

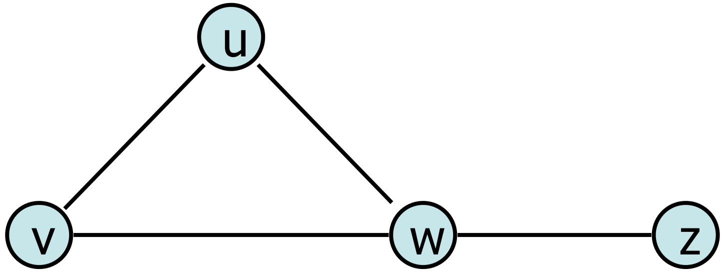
Learning Objectives

Review adjacency matrix graph implementations

Introduce adjacency list implementation

Discuss the strengths and weaknesses of each implementation

Graph Implementation: Adjacency Matrix



U	0
V	1
W	2
Z	3

	u	v	w	z
u	0	1	1	0
v	1	0	1	0
w	1	1	0	1
z	0	0	1	0

Vertex Storage:

A dictionary (or list) of vertices

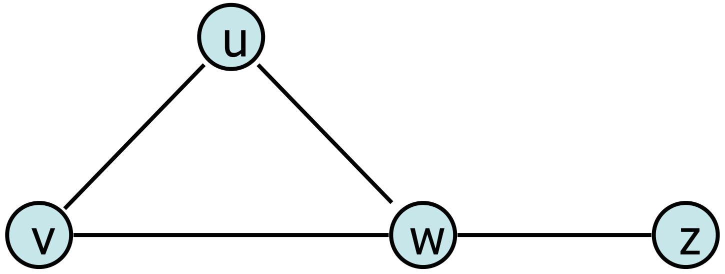
Also stores the indexing of the matrix

Edge Storage:

A |v| x |v| matrix storing edges

e[r][c] = 1 if there is an edge between r and c

Graph Implementation: Adjacency Matrix



U	0
V	1
W	2
Z	3

	u	v	w	z
u	0	1	1	0
v	1	0	1	0
w	1	1	0	1
z	0	0	1	0

Pros:

Fast lookup and modification of edges

Easily extended to weighted and directed

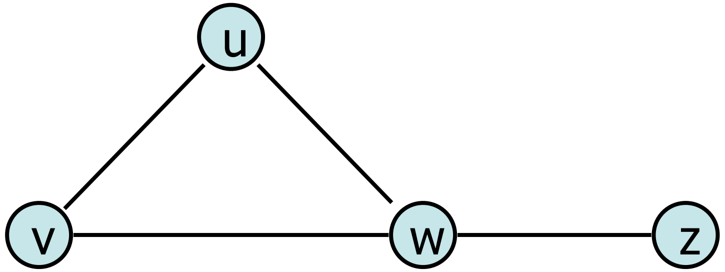
Cons:

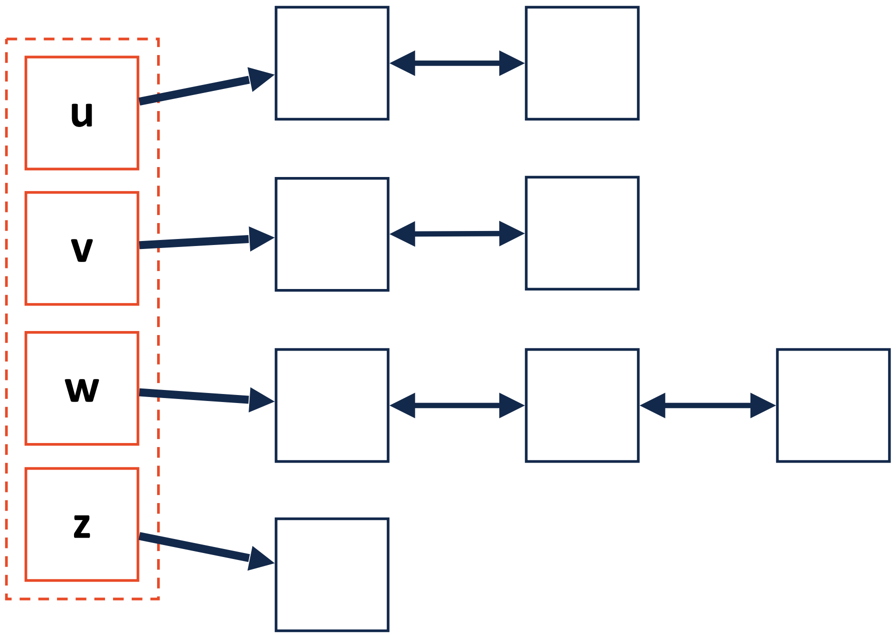
Slow to add or remove new vertices

Getting the degree of nodes is slow

Storage costs are relatively large

Adjacency List

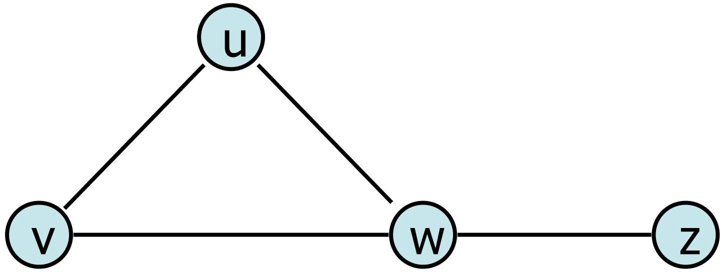


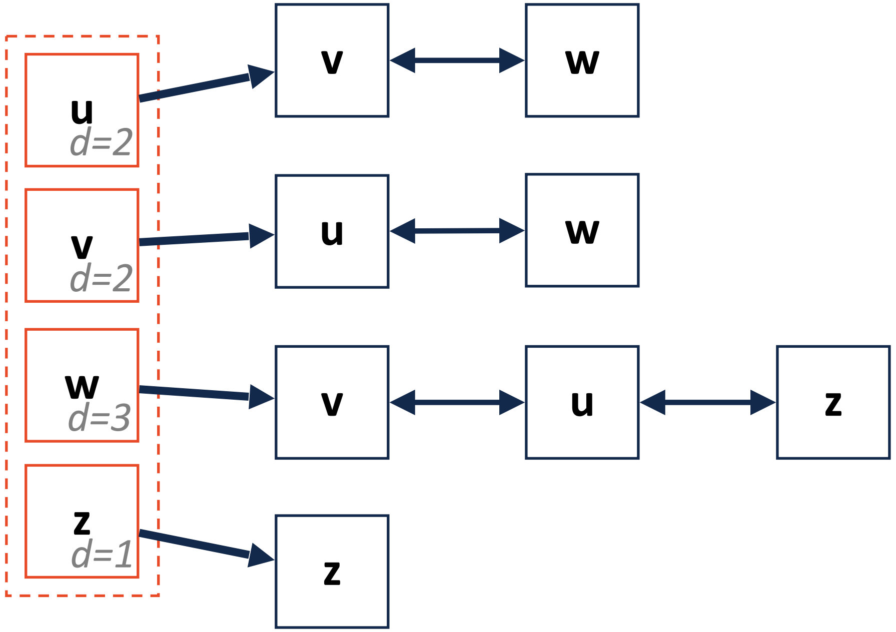


Vertex Storage:

Edge Storage:

Adjacency List

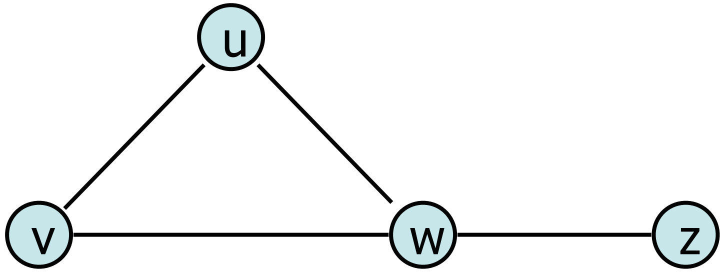




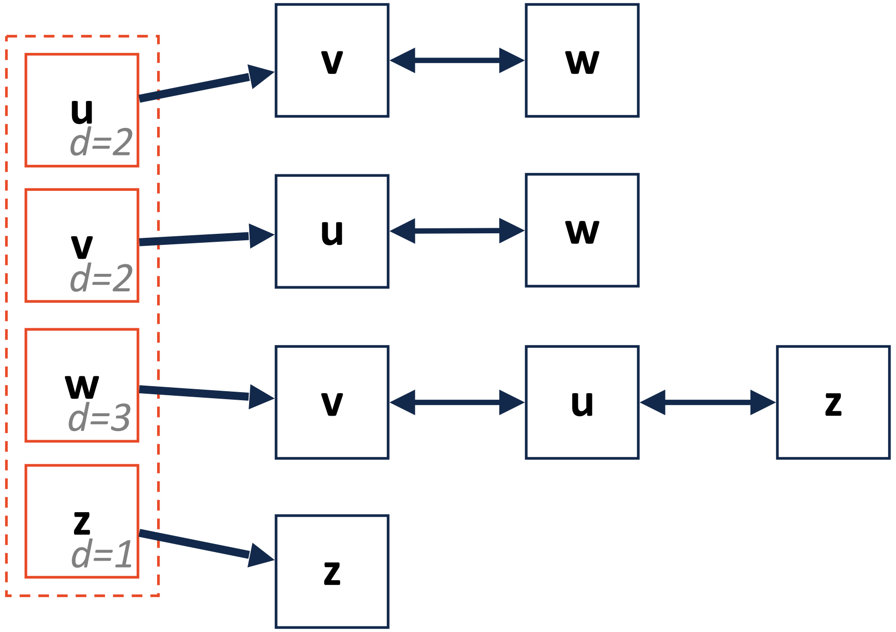
Vertex Storage:

Edge Storage:

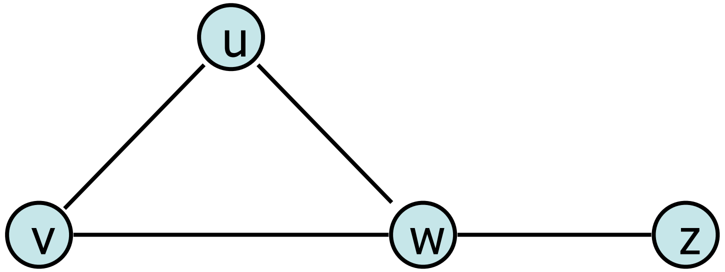
Adjacency List



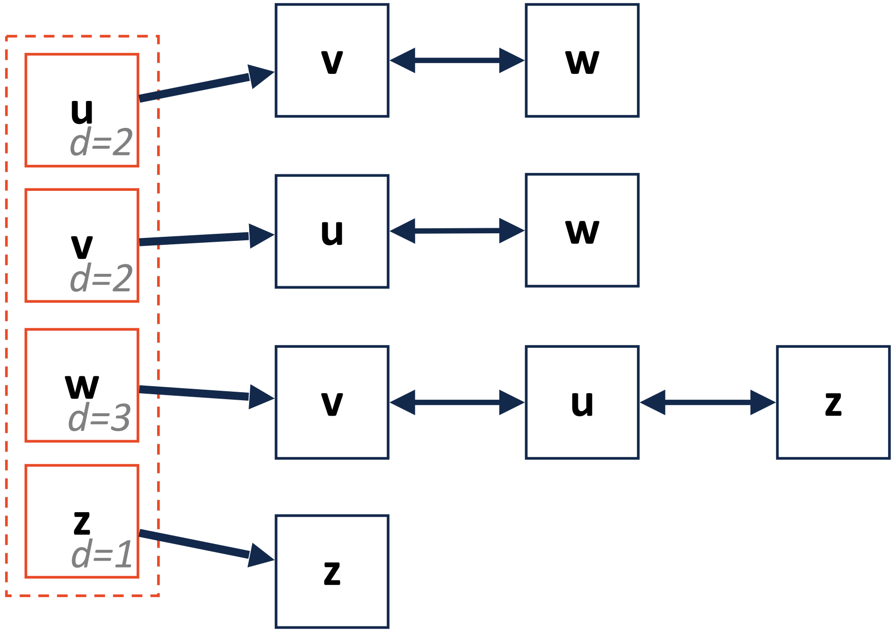
getVertices():



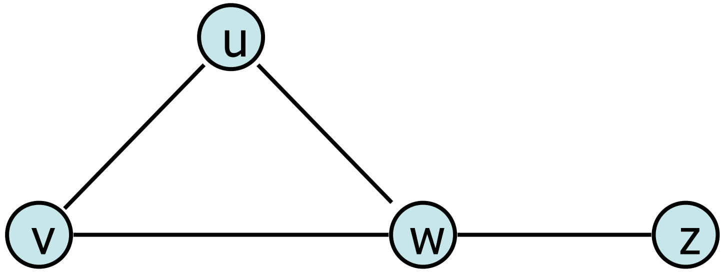
Adjacency List

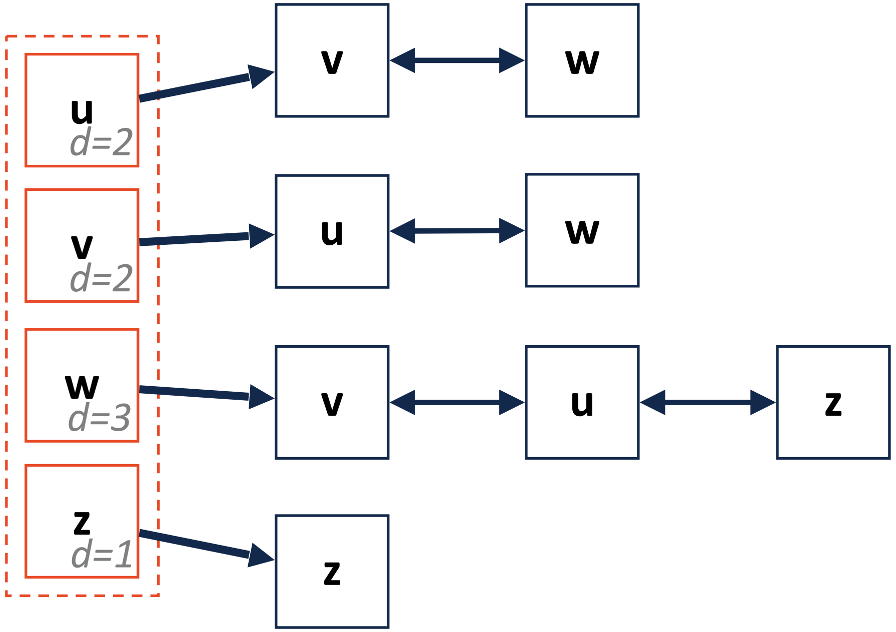


getEdges(v):



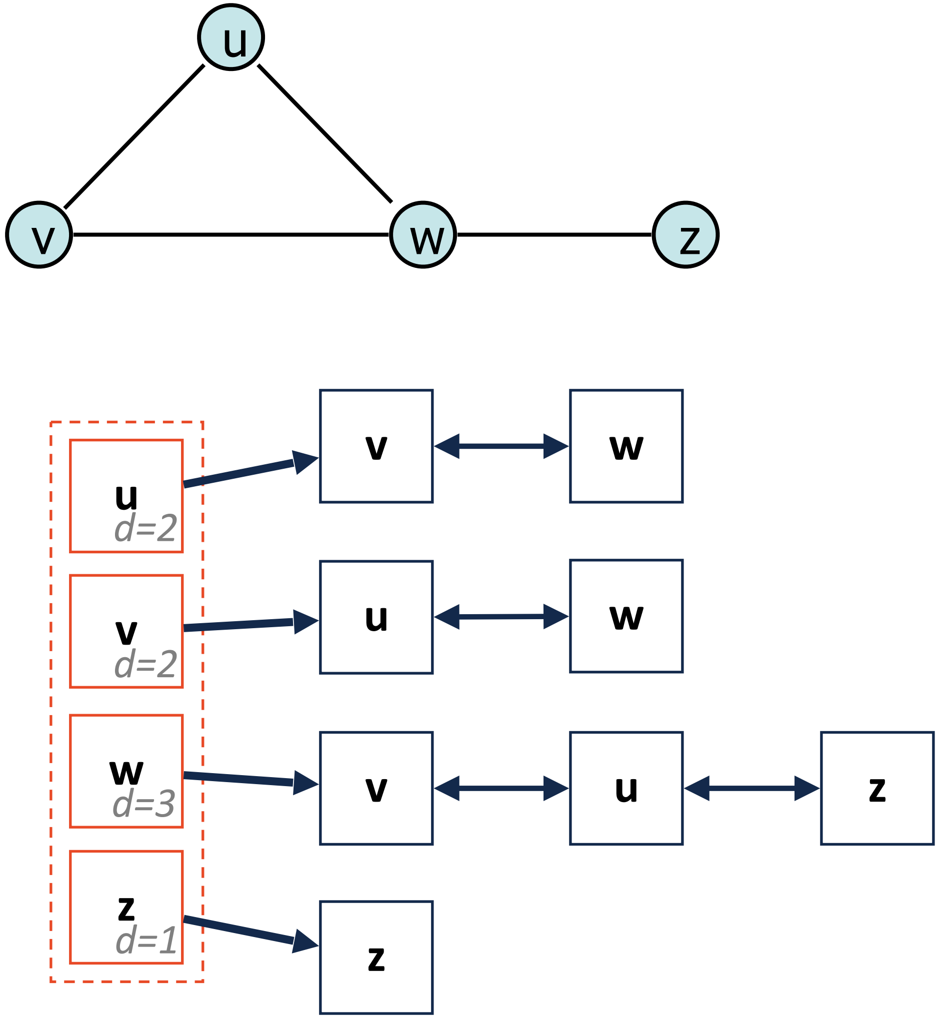
Adjacency List





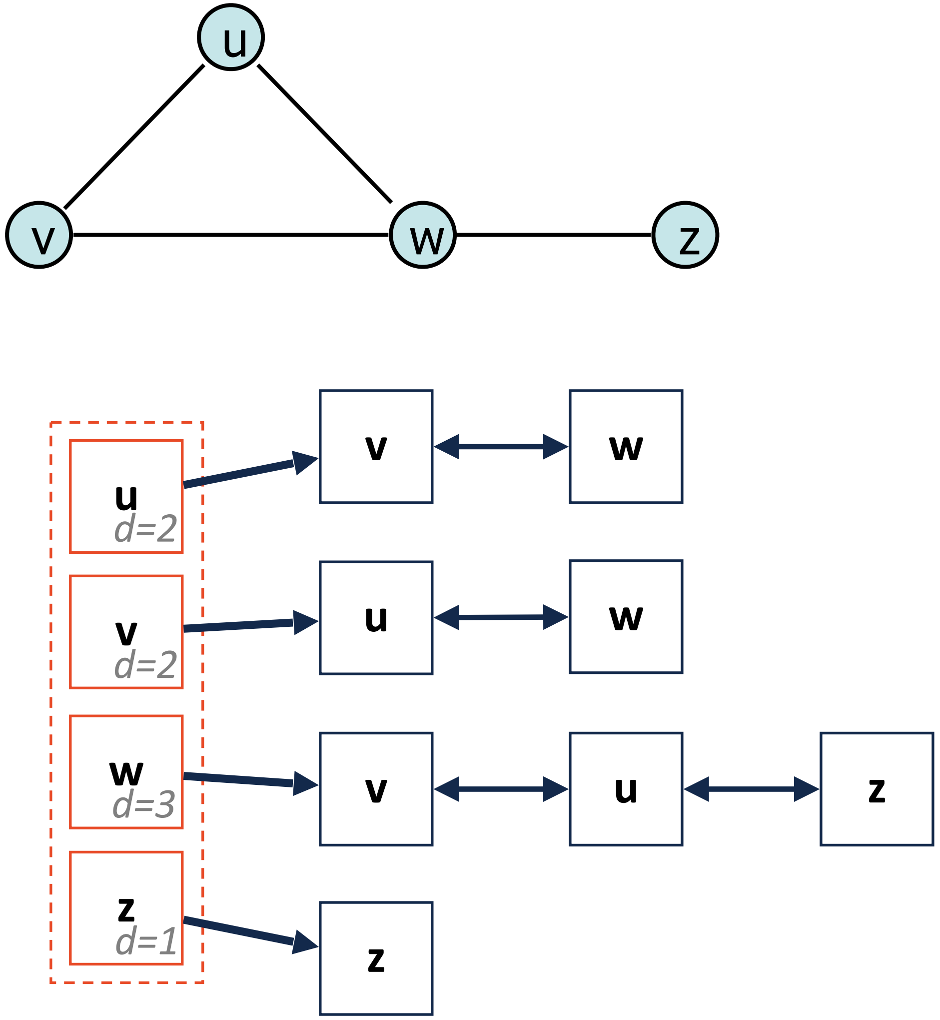
areAdjacent(u, v):

Adjacency List



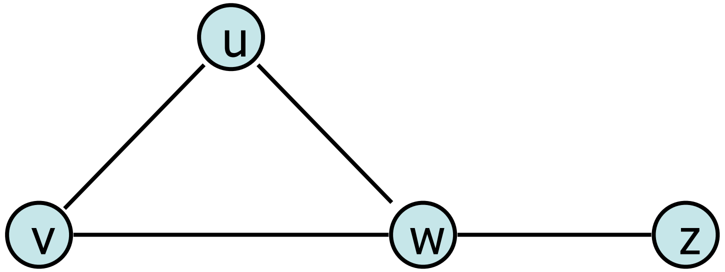
insertVertex(v):

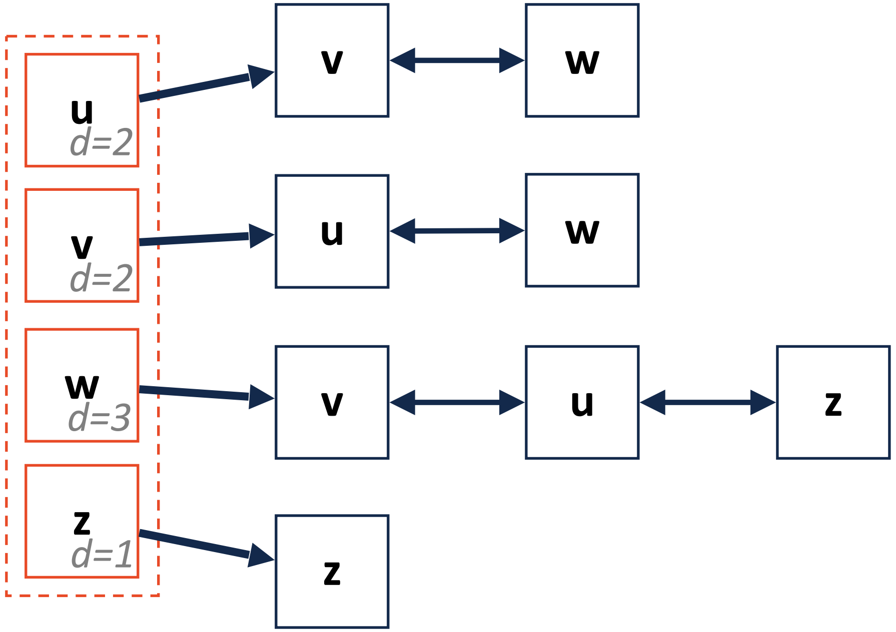
Adjacency List



removeVertex(v):

Adjacency List

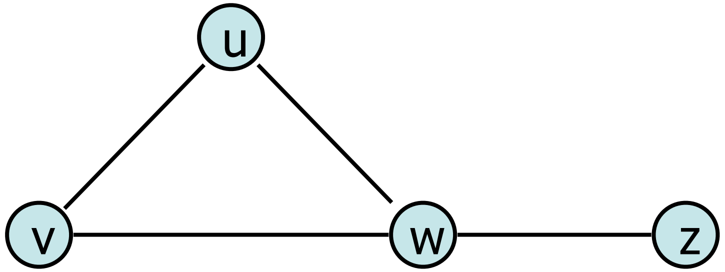


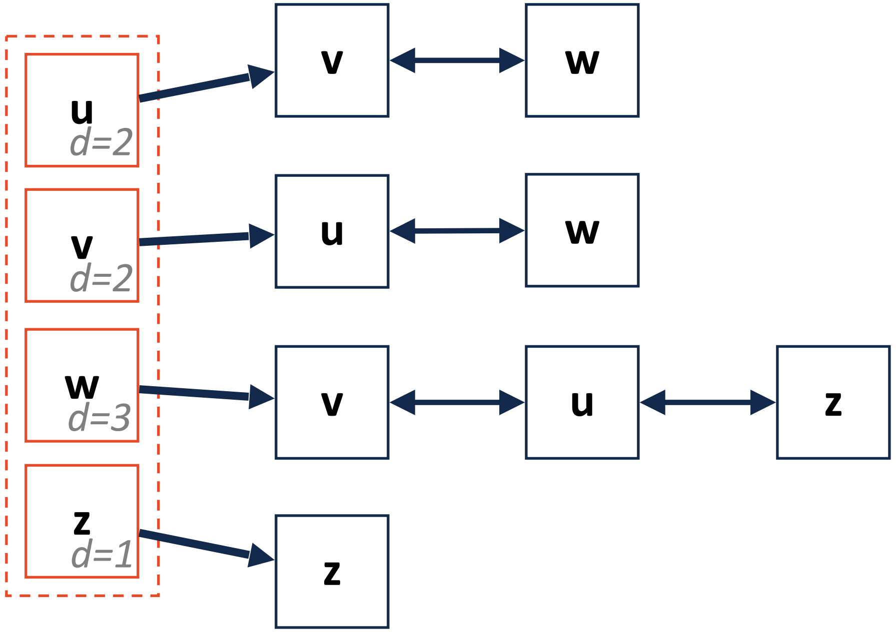


insertEdge(u, v):

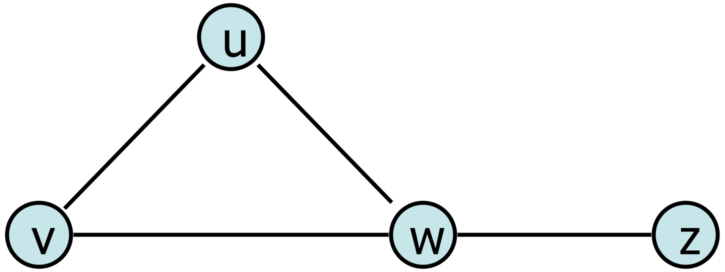
Adjacency List

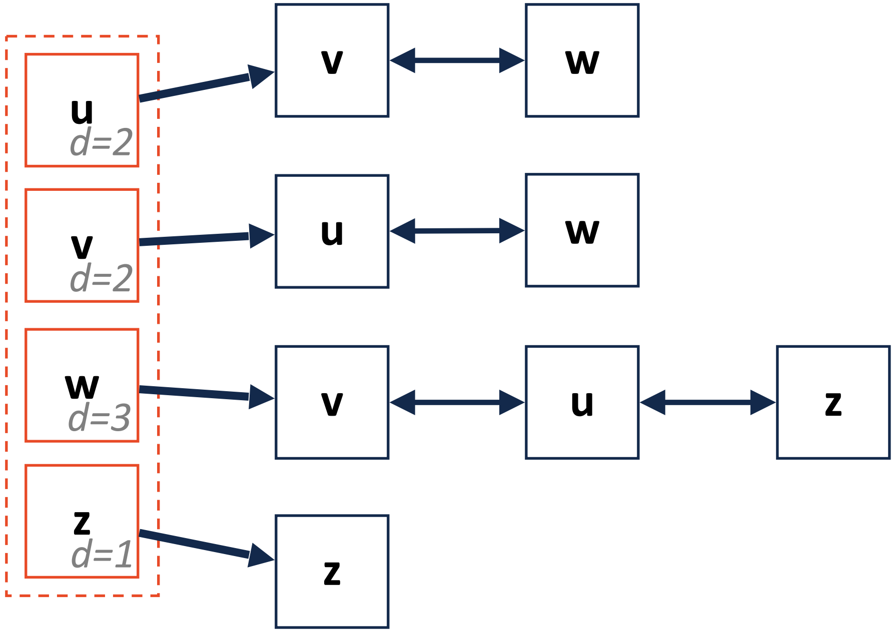
removeEdge(u, v):





Adjacency List



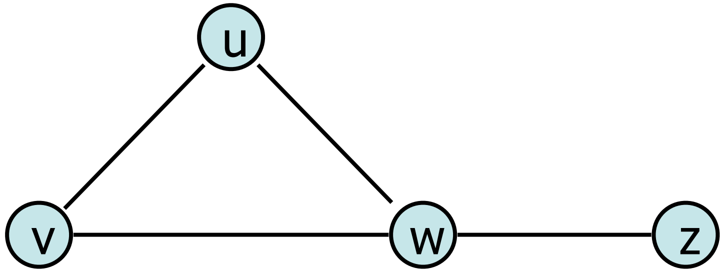


Pros:

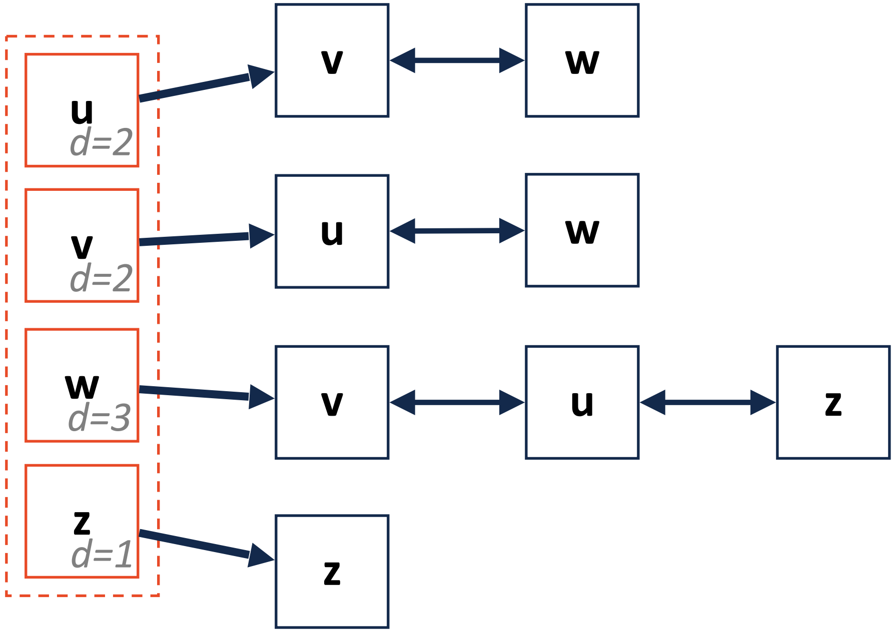
Cons:

Adjacency List

How would our data structure change if…

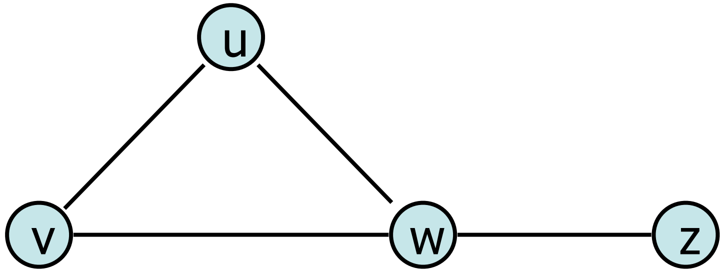


Edges are directed:

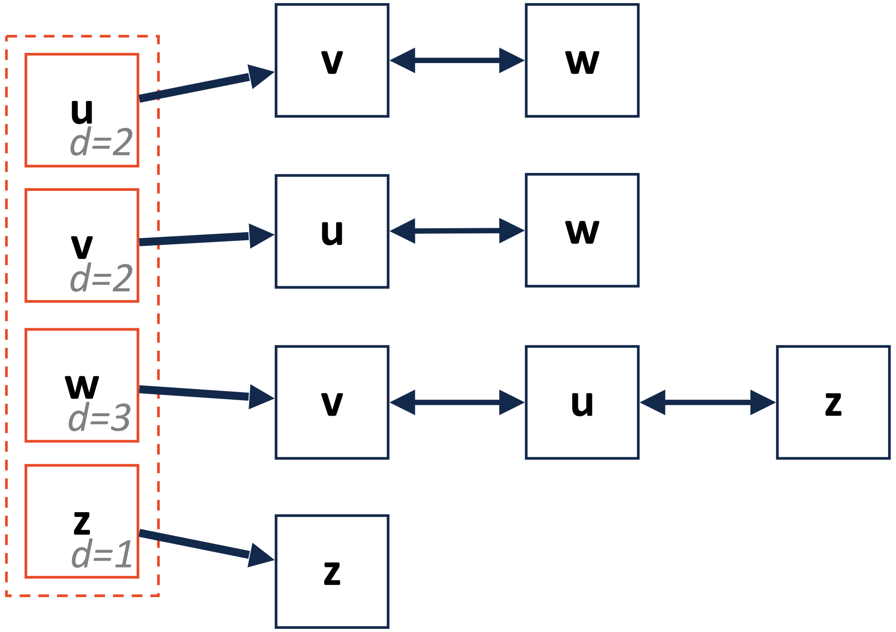


Adjacency List

How would our data structure change if…



Edges are weighted:



$|V| = n, |E| = m$

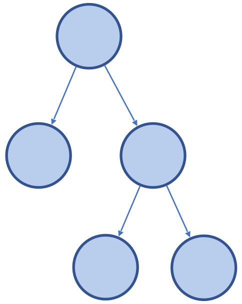
Expressed as O(f)	Edge List	Adjacency Matrix	Adjacency List
Space	n+m	n^2	n+m
insertVertex(v)	1*	n*	1*
removeVertex(v)	m**	n	deg(v)***
insertEdge(u, v)	1	1	1*
removeEdge(u, v)	m	1	min(deg(u), deg(v))
getEdges(v)	m	n	deg(v)
areAdjacent(u, v)	m	1	min(deg(u), deg(v))

Graph Traversals

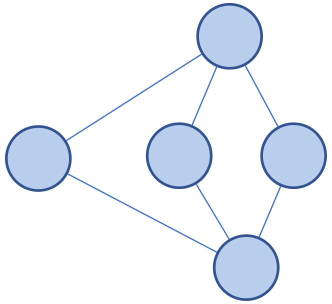
There is no clear order in a graph (even less than a tree!)

How can we systematically go through a complex graph in the fewest steps?

Tree traversals won't work — lets compare:

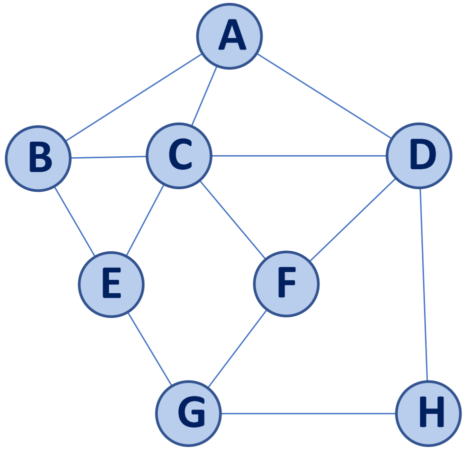


- Rooted
- Acyclic
-

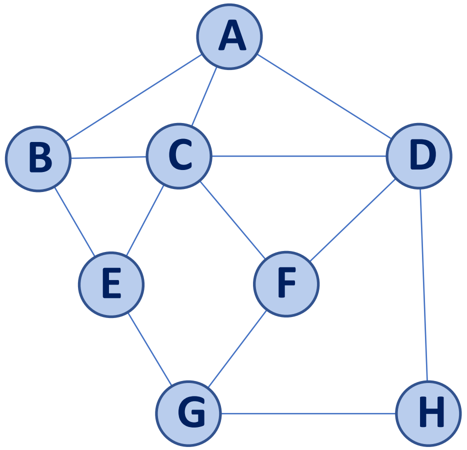


-
-
-

Traversal: BFS

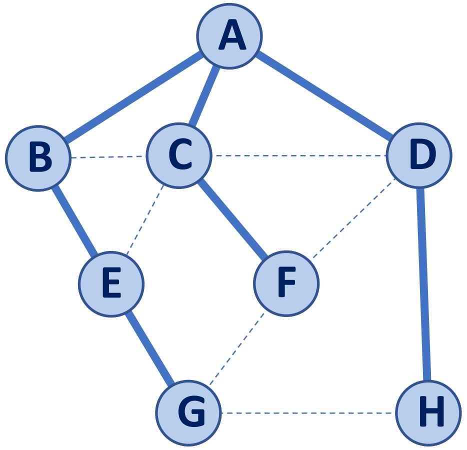


Traversal: BFS

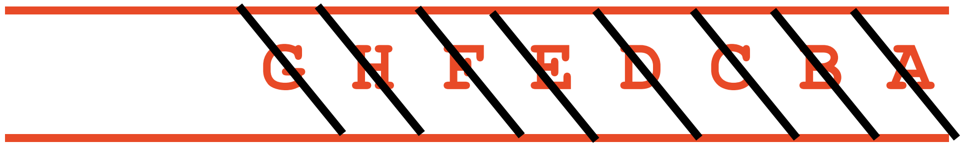


v	d	P	Adjacent Edges
A			
B			
C			
D			
E			
F			
G			
H			

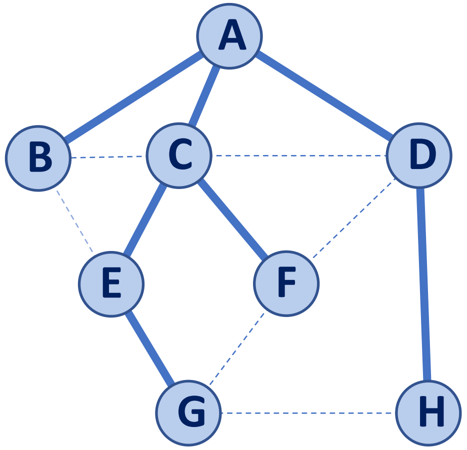
Traversal: BFS



v	d	P	Adjacent Edges
A	0	-	B C D
B	1	A	A C E
C	1	A	A B D E F
D	1	A	A C F H
E	2	B	B C G
F	2	C	C D G
G	3	E	E F H
H	2	D	D G



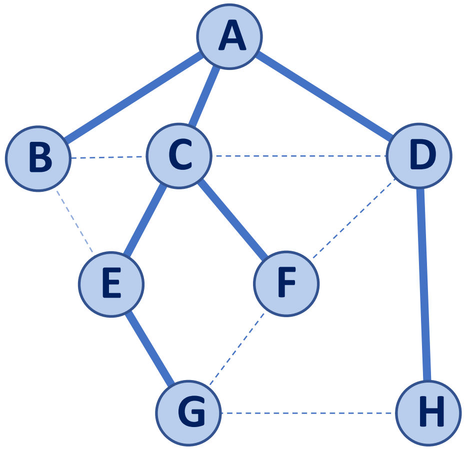
Traversal: BFS



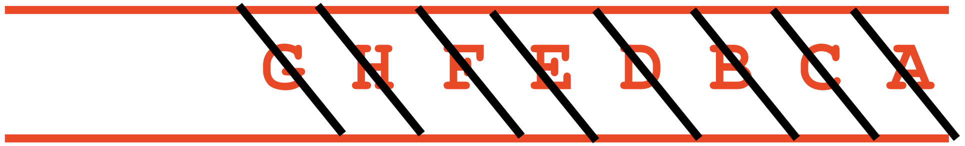
v	d	P	Adjacent Edges
A	0	-	C B D
B	1	A	A C E
C	1	A	A B D E F
D	1	A	A C F H
E	2	C	B C G
F	2	C	C D G
G	3	E	E F H
H	2	D	D G

~~G~~ ~~H~~ ~~F~~ ~~E~~ ~~D~~ ~~B~~ ~~C~~ ~~A~~

Running time of BFS



v	d	P	Adjacent Edges
A	0	-	C B D
B	1	A	A C E
C	1	A	B A D E F
D	1	A	A C F H
E	2	C	B C G
F	2	C	C D G
G	3	E	E F H
H	2	D	D G



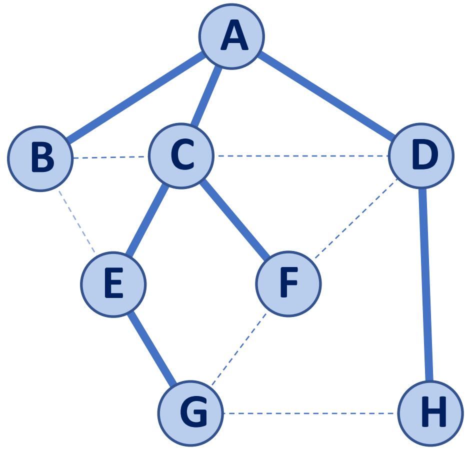
BFS Observations

What is the shortest path from **A** to **H**?

What is the shortest path from **E** to **H**?

If my node has distance **d**, do I know anything about the nodes connected by a **cross edge**?

v	d	P	Adjacent Edges
A	0	-	C B D
B	1	A	A C E
C	1	A	B A D E F
D	1	A	A C F H
E	2	C	B C G
F	2	C	C D G
G	3	E	E F H
H	2	D	D G



BFS Observations

BFS can be used to detect cycles

The value of d in BFS is the shortest distance from source to every vertex

In BFS, the endpoints of a cross edge never differ in distance, d, by more than 1. In other words for vertices u and v connected by a cross edge: $\left(|d(u) - d(v)| \leq 1 \right)$