Algorithms and Data Structures for Data Science
Graphs

CS 277
Brad Solomon

November 1, 2021

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Exam 2

Average (before partial credit):

Final Project Mid-Project Check-ins

Mid-project check-ins start next week (November 8th-12th)!

Proposal resubmissions due November 5th

To sign up for a mid-project meeting time:

https://bit.ly/3GzmC3P

Learning Objectives

Define graph vocabulary

Discuss graph implementation / storage strategies

Define key graph functions and discuss implementation details

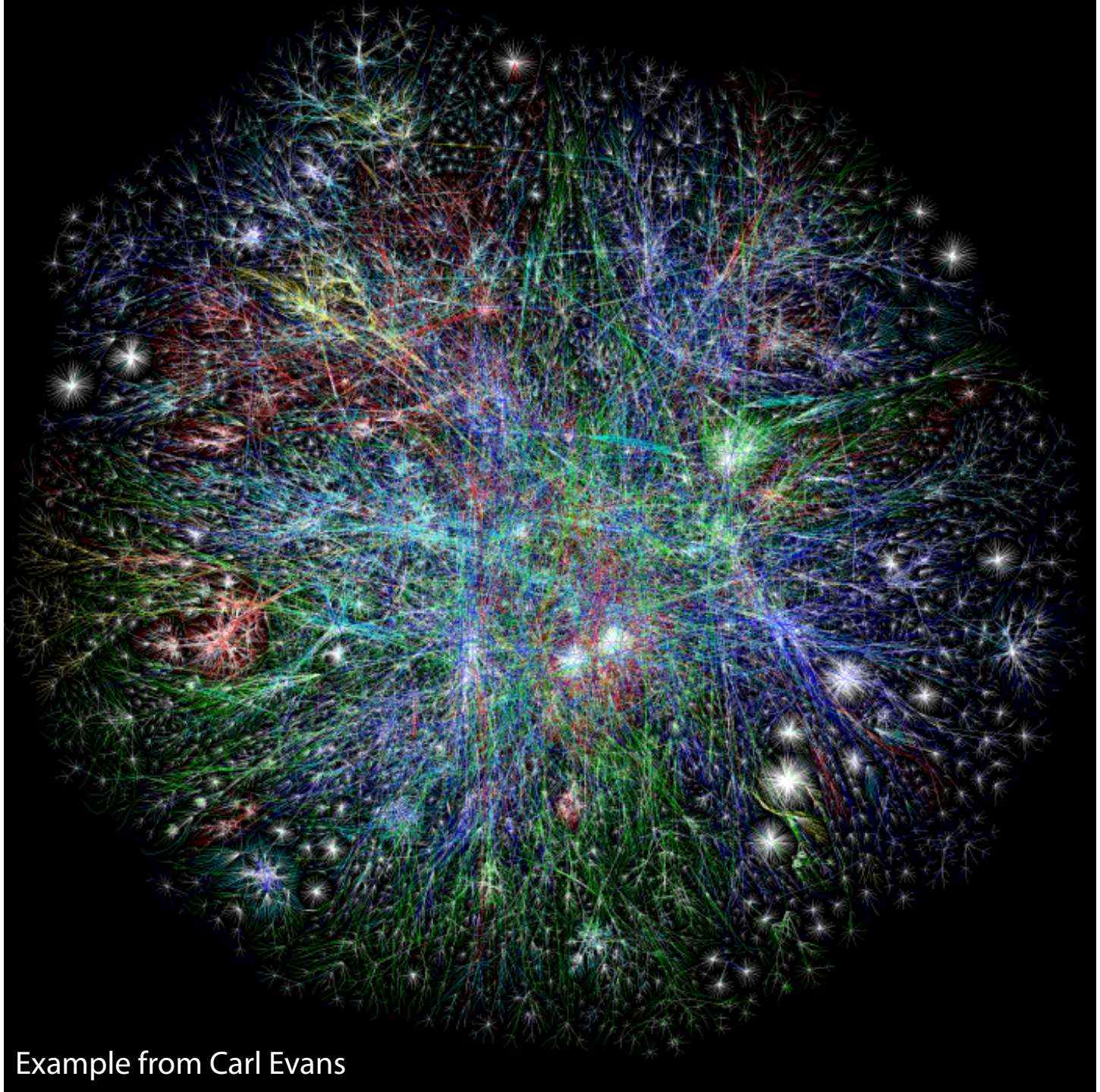
In Review: Data Structures

Array

- Sorted Array
- Unsorted Array
  - Stacks
  - Queues

Linked

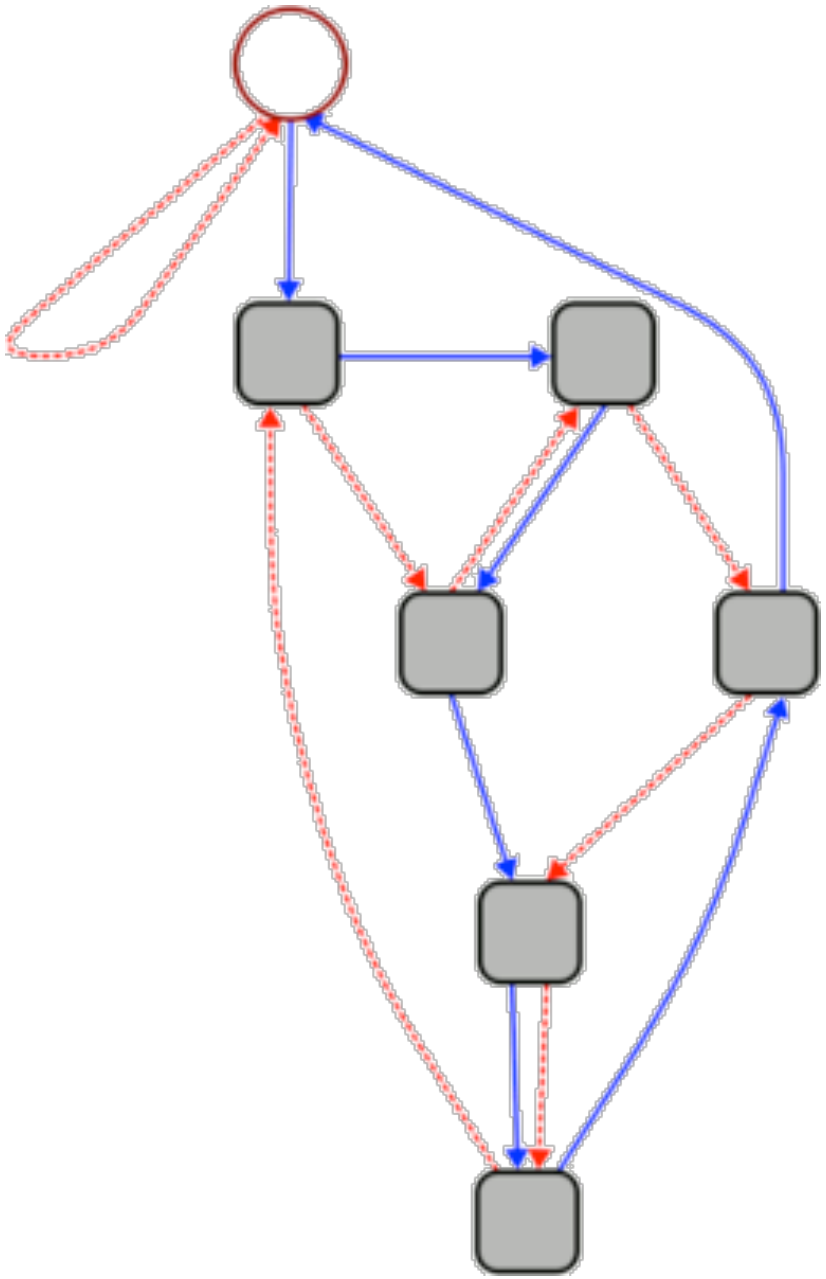
- Linked List
- Trees
  - Binary Tree
    - Huffman Encoding
    - BST
    - AVL Tree



The Internet 2003

The OPTE Project (2003)

Map of the entire internet; nodes are routers; edges are connections.



This graph can be used to quickly calculate whether a given number is divisible by 7.

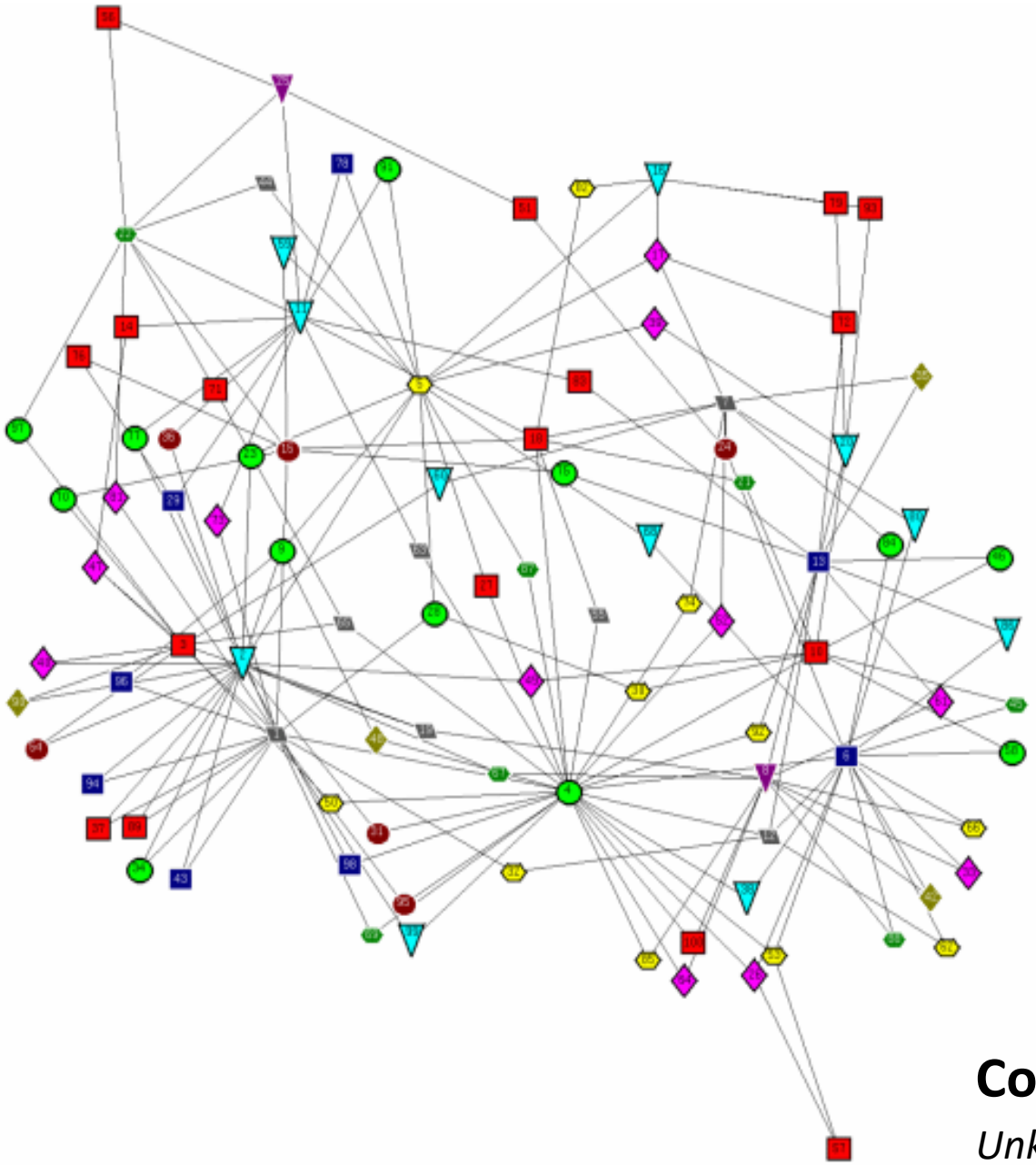
1. Start at the circle node at the top.
2. For each digit **d** in the given number, follow **d** blue (solid) edges in succession. As you move from one digit to the next, follow **1** red (dashed) edge.
3. If you end up back at the circle node, your number is divisible by 7.

3703

“Rule of 7”

Unknown Source

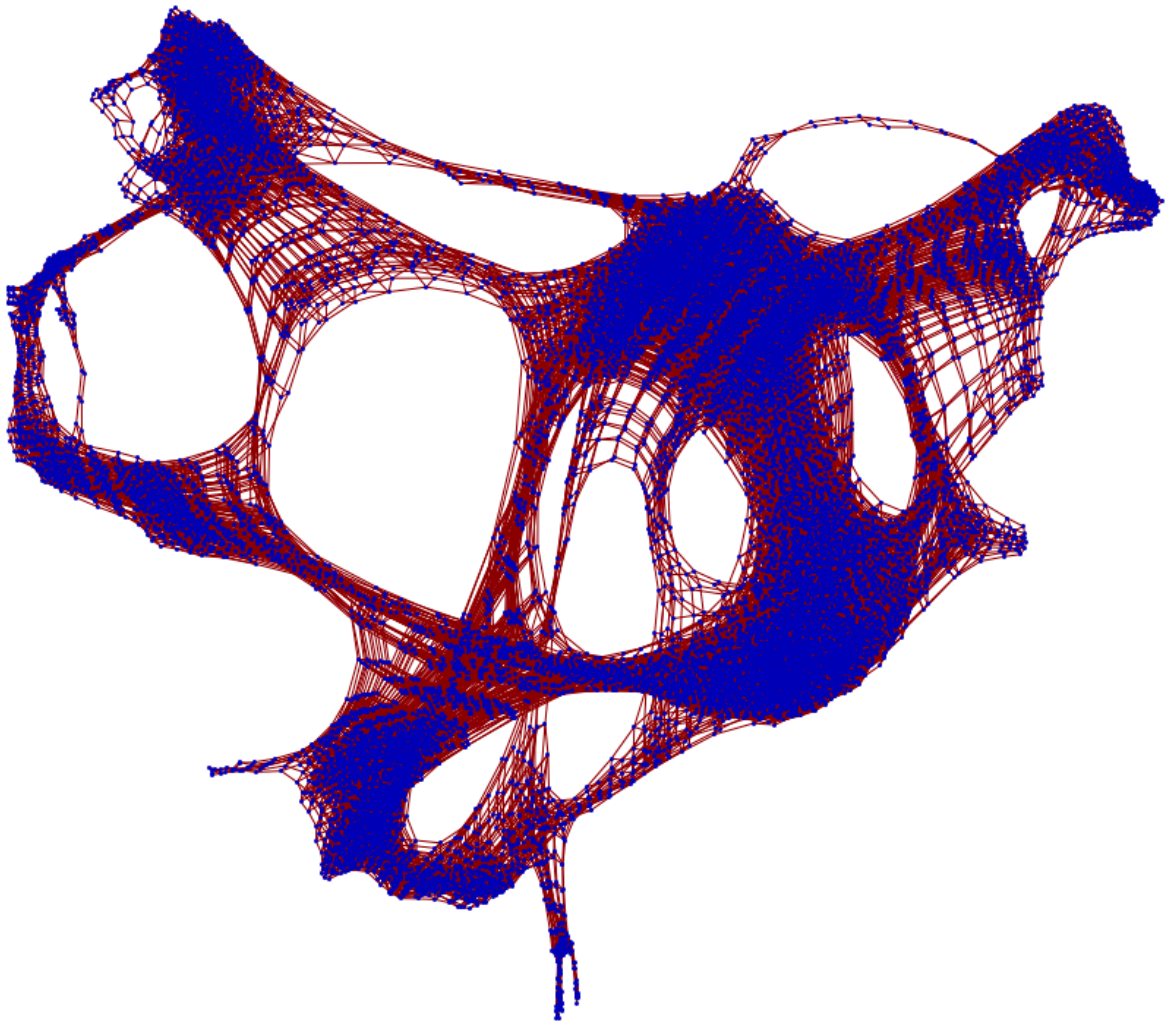
Presented by Cinda Heeren, 2016



Conflict-Free Final Exam Scheduling Graph

Unknown Source

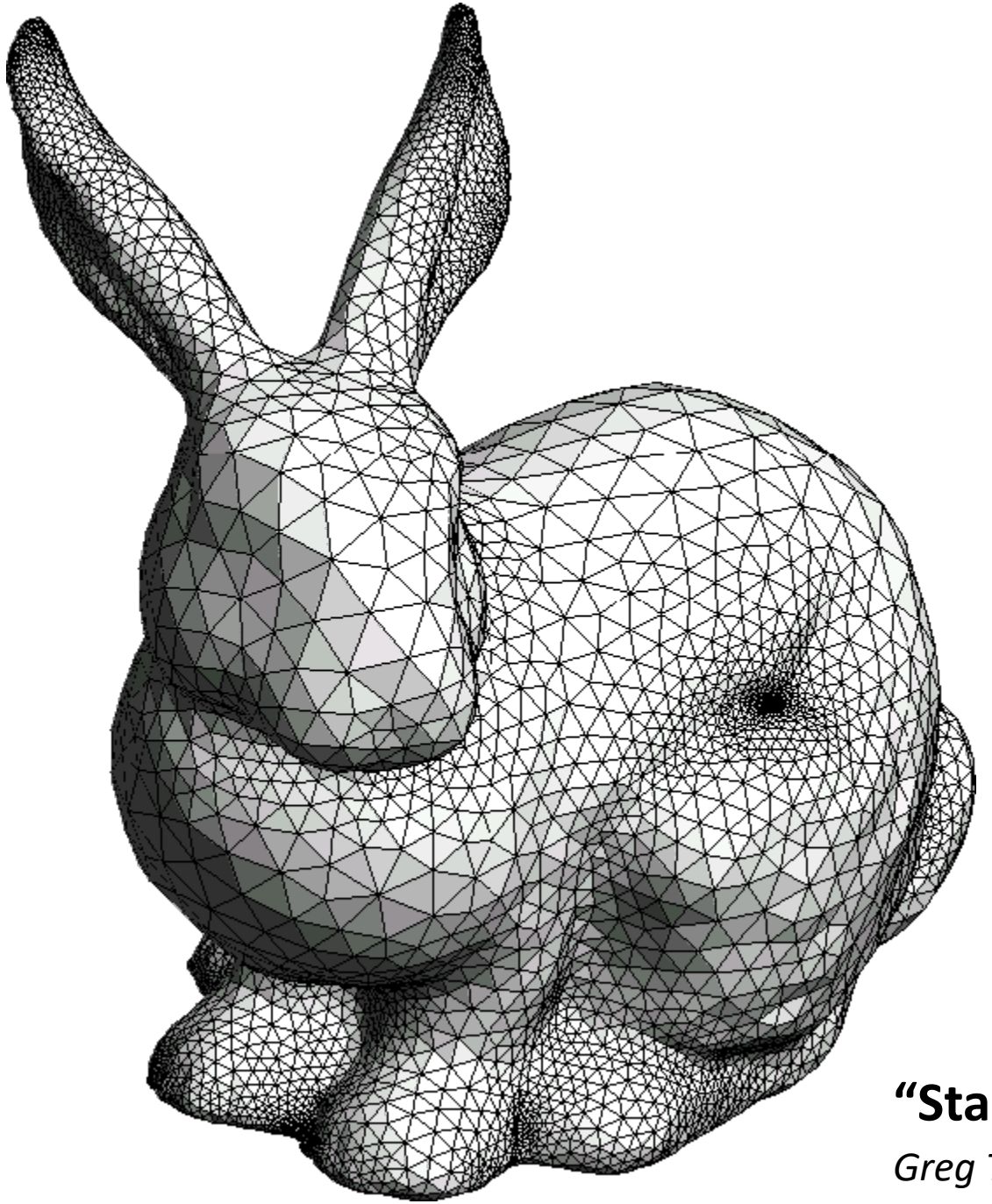
Presented by Cinda Heeren, 2016



“Rush Hour” Solution

Figure from Christopher Wolfram

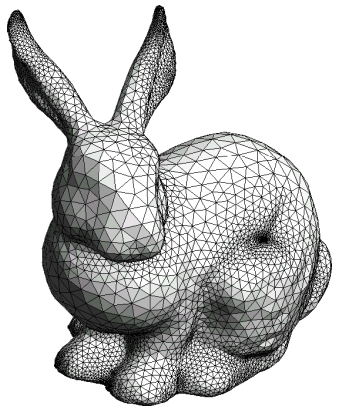
Idea from Cinda Heeren



“Stanford Bunny”

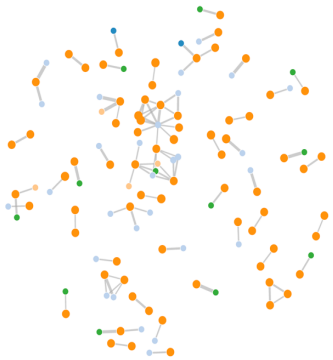
Greg Turk and Mark Levoy (1994)

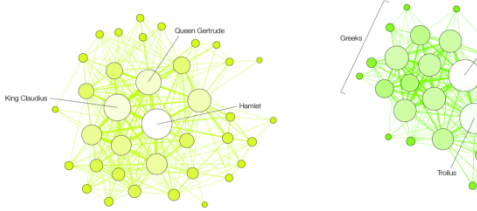
Graphs



To study all of these structures:

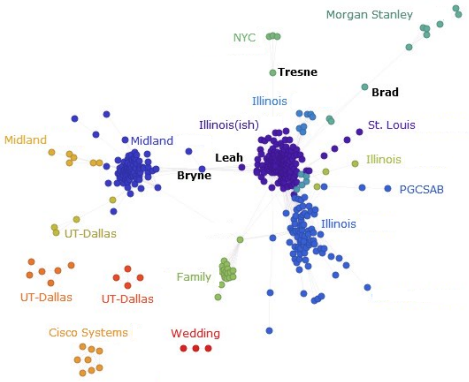
1. A common vocabulary
2. Graph implementations
3. Graph traversals
4. Graph algorithms

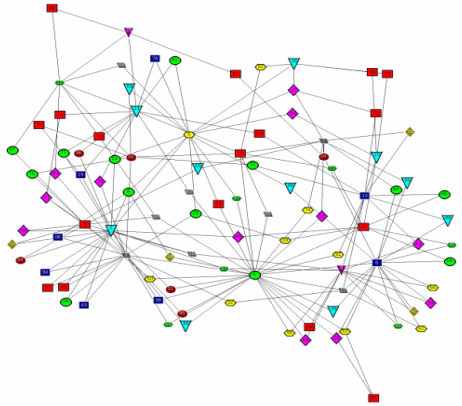


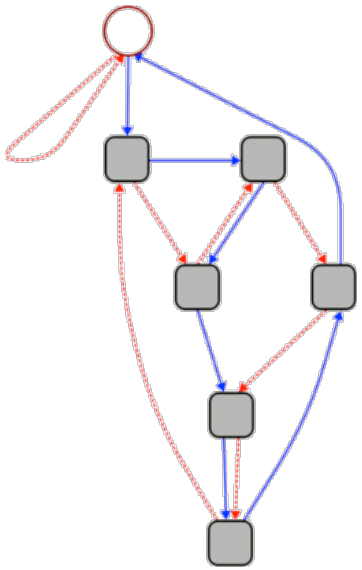


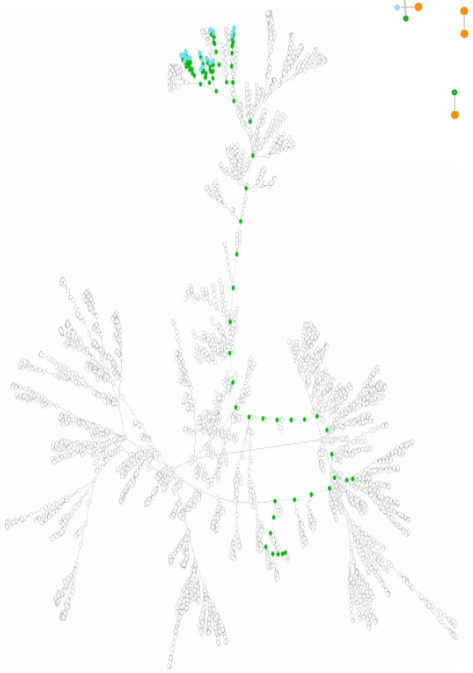
HAMLET

TROILUS AND CRESSIDA



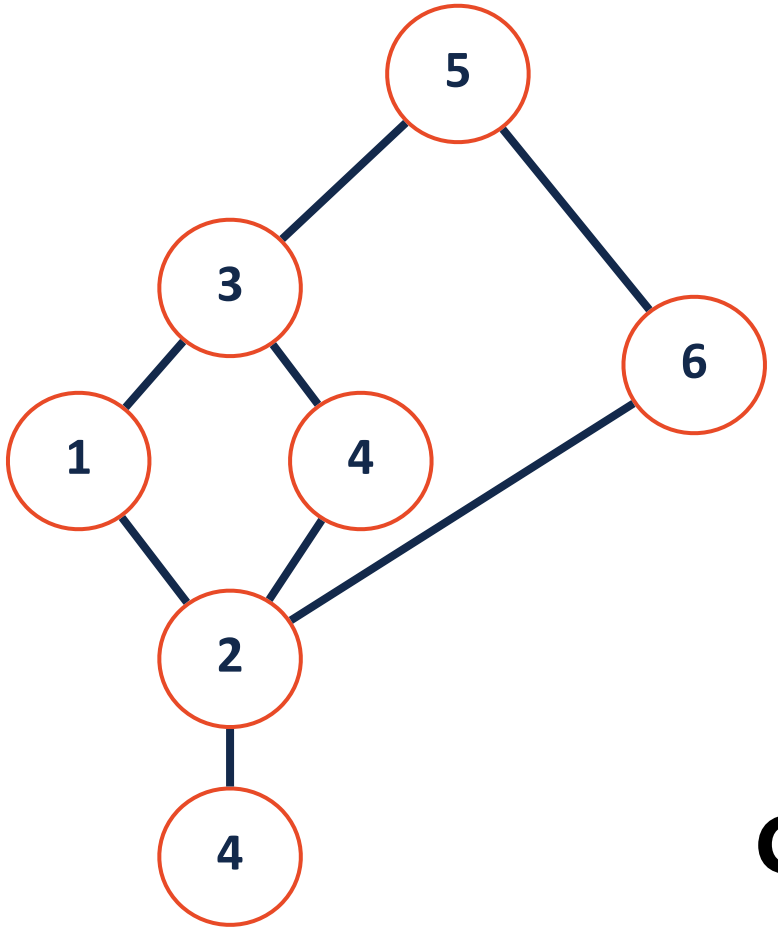






Graph Vocabulary

$G = (V, E)$



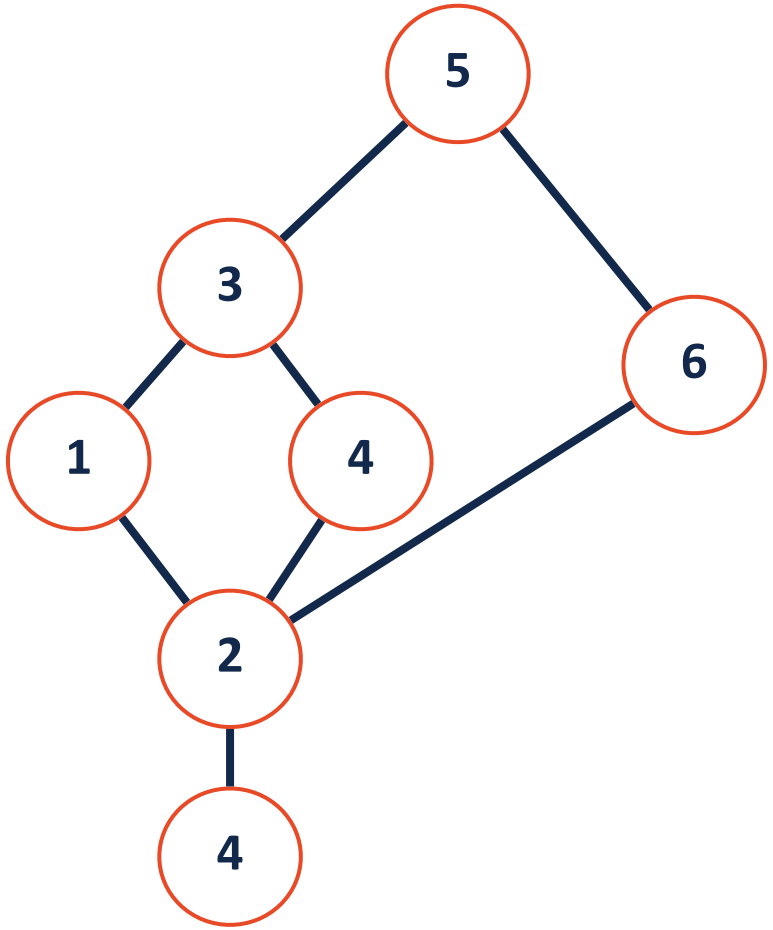
Vertex:

Edges:

Cardinality:

Graph Vocabulary

$G = (V, E)$



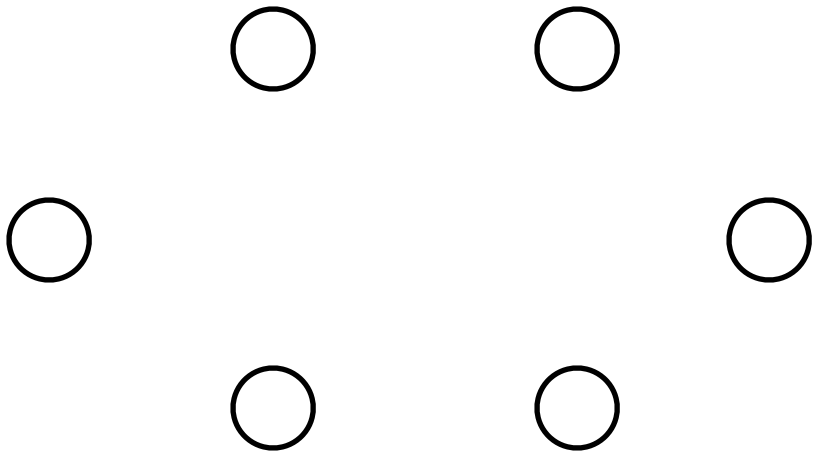
Degree:

Adjacent:

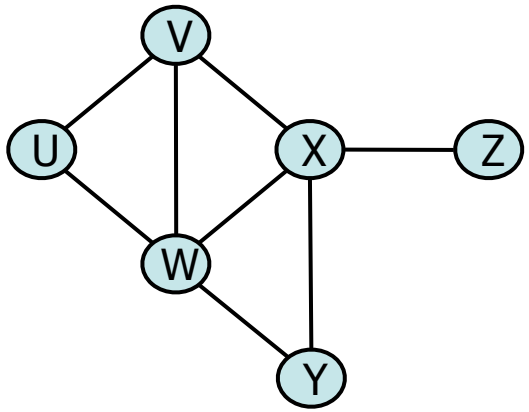
Path:

Cycle:

Connected Graphs



Graph Efficiency



Graph efficiency has two factors:

n, the number of vertices

m, the number of edges.

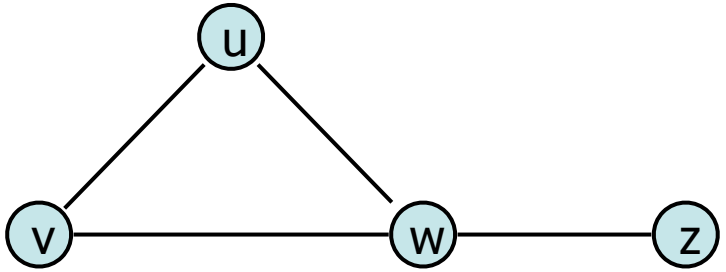
Minimum Edges:

Maximum Edges:

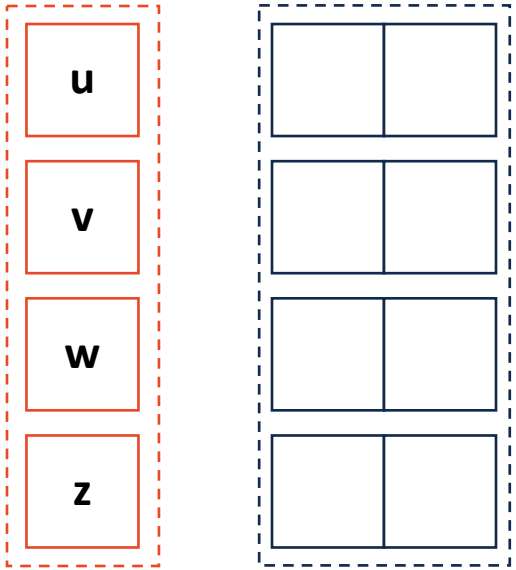
$$\sum_{v \in V} deg(v) =$$

Graph Implementation: Edge List

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



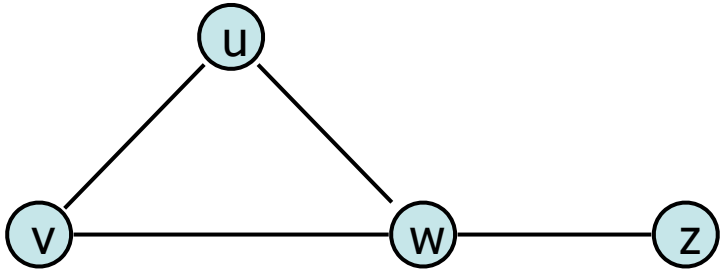
Vertex Storage:



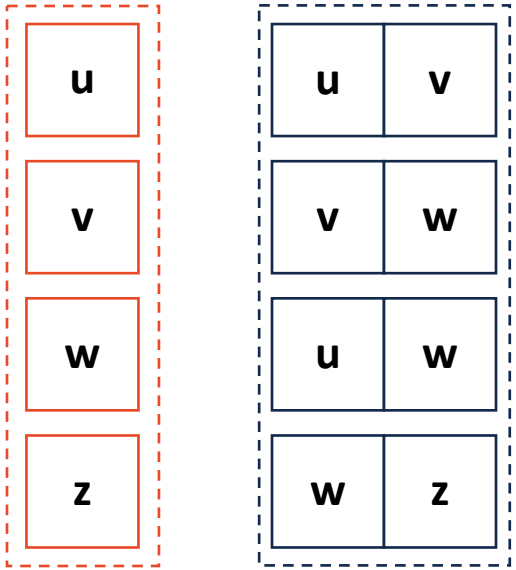
Edge Storage:

Graph Implementation: Edge List

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



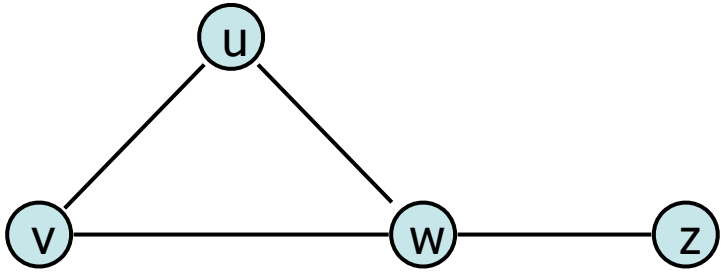
insertVertex(v):



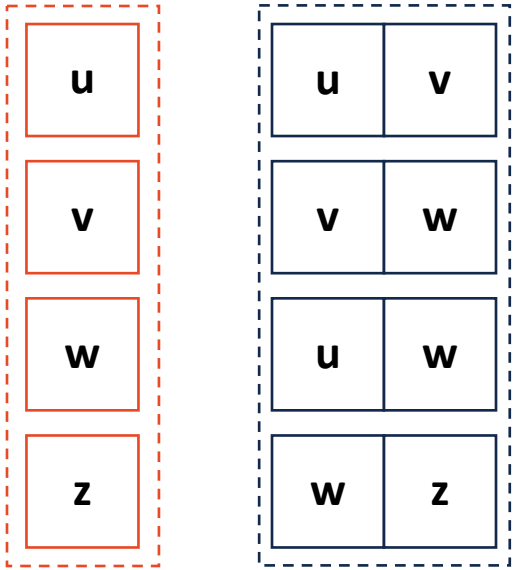
removeVertex(v):

Graph Implementation: Edge List

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



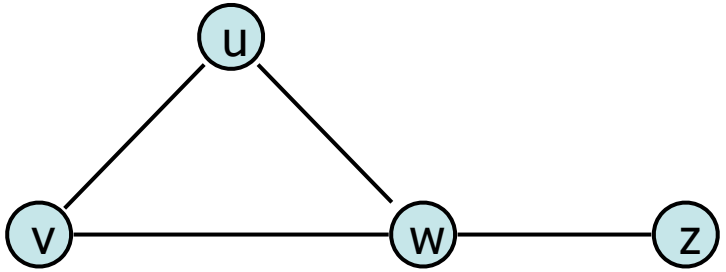
insertEdge(u, v):



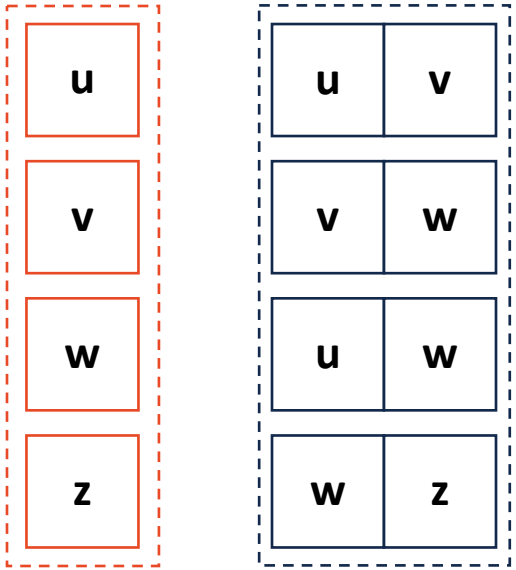
removeEdge(u, v):

Graph Implementation: Edge List

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



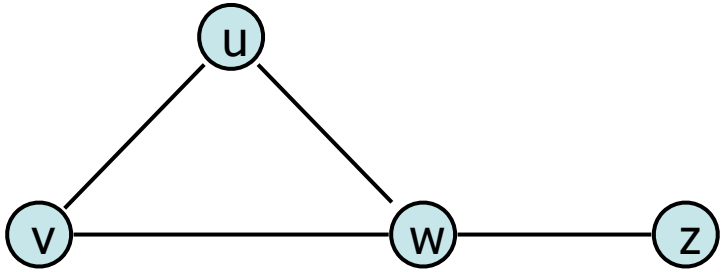
getEdges(u):

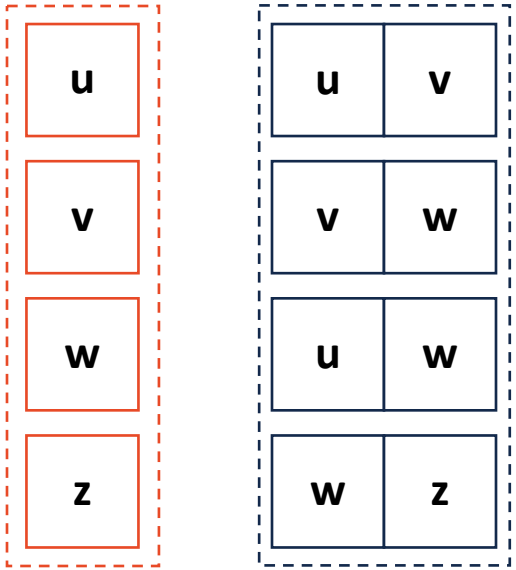


areAdjacent(u, v):

Graph Implementation: Edge List

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

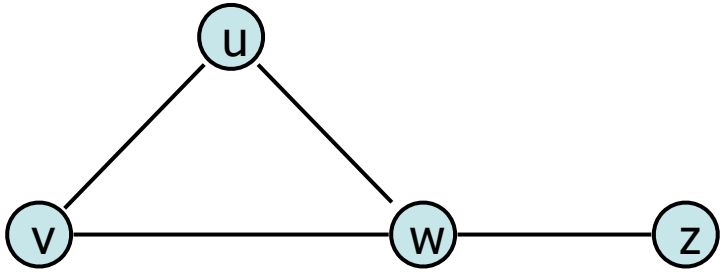




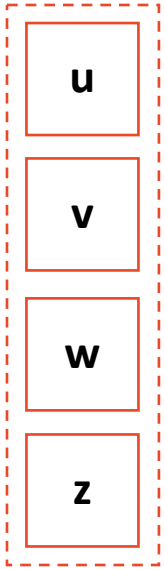
Pros:

Cons:

Graph Implementation: Adjacency Matrix



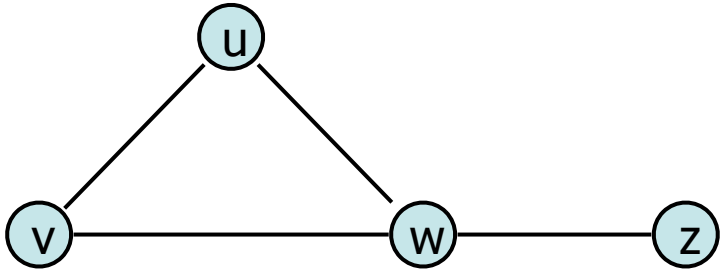
Vertex Storage:



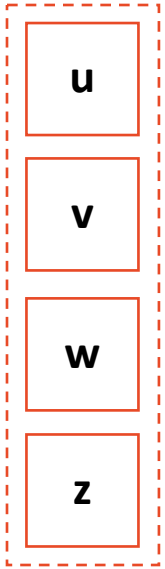
Edge Storage:

	u	v	w	z
u				
v				
w				
z				

Graph Implementation: Adjacency Matrix

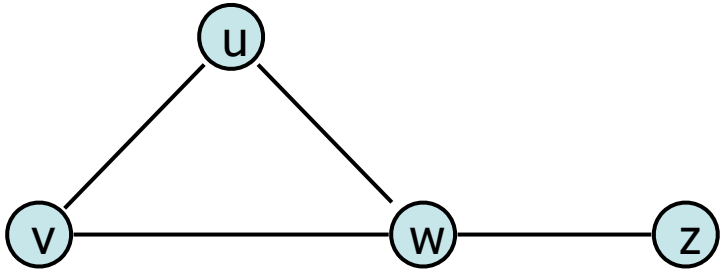


insertVertex(v):

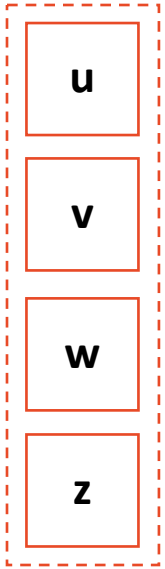


	u	v	w	z
u	-	1	1	0
v		-	1	0
w			-	1
z				-

Graph Implementation: Adjacency Matrix

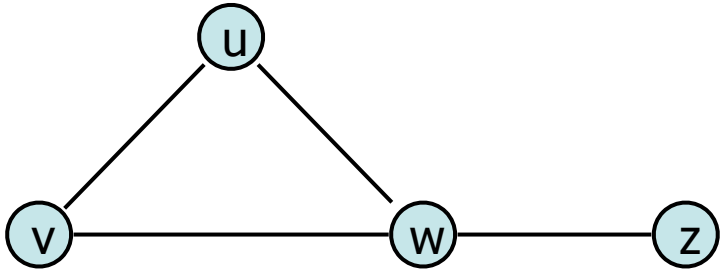


removeVertex(v):

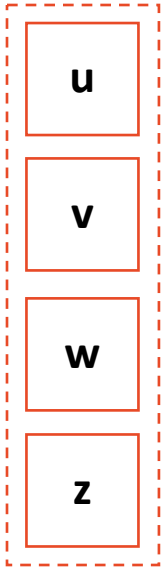


	u	v	w	z
u	-	1	1	0
v		-	1	0
w			-	1
z				-

Graph Implementation: Adjacency Matrix

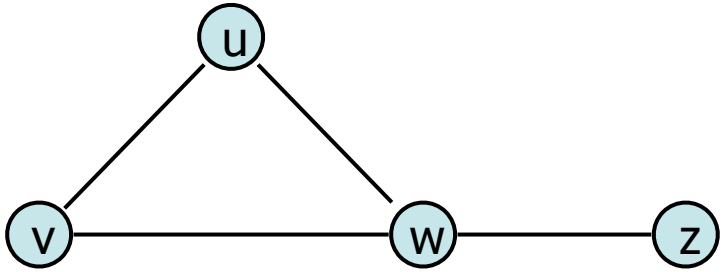


insertEdge(u, v):

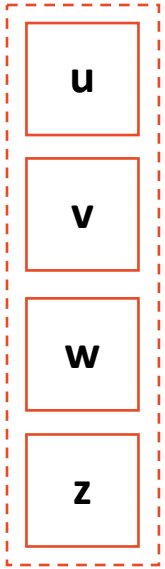


	u	v	w	z
u	-	1	1	0
v		-	1	0
w			-	1
z				-

Graph Implementation: Adjacency Matrix

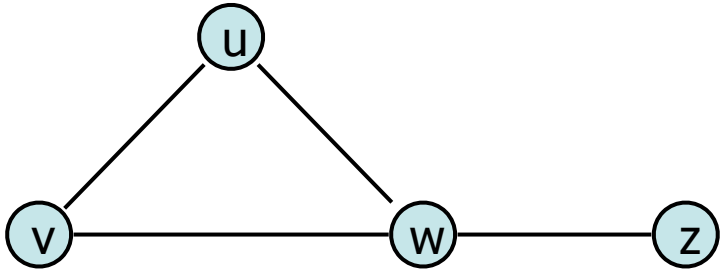


removeEdge(u, v):

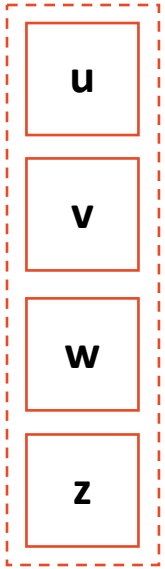


	u	v	w	z
u	-	1	1	0
v		-	1	0
w			-	1
z				-

Graph Implementation: Adjacency Matrix

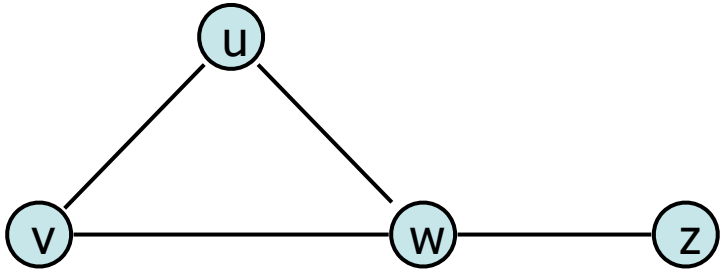


getEdges(u):

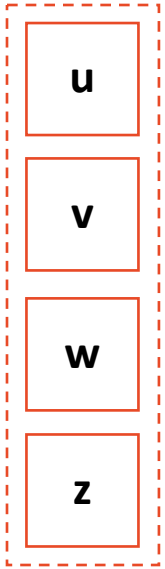


	u	v	w	z
u	-	1	1	0
v		-	1	0
w			-	1
z				-

Graph Implementation: Adjacency Matrix

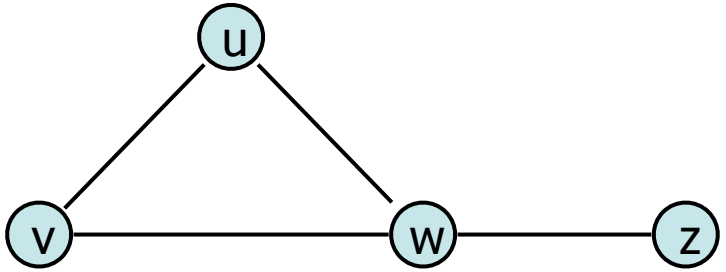


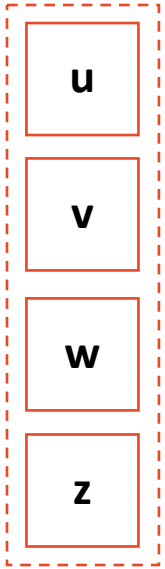
areAdjacent(u, v):



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

Graph Implementation: Adjacency Matrix



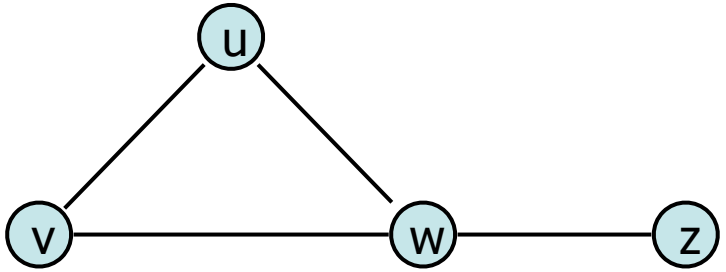


	u	v	w	z
u	-	1	1	0
v		-	1	0
w			-	1
z				-

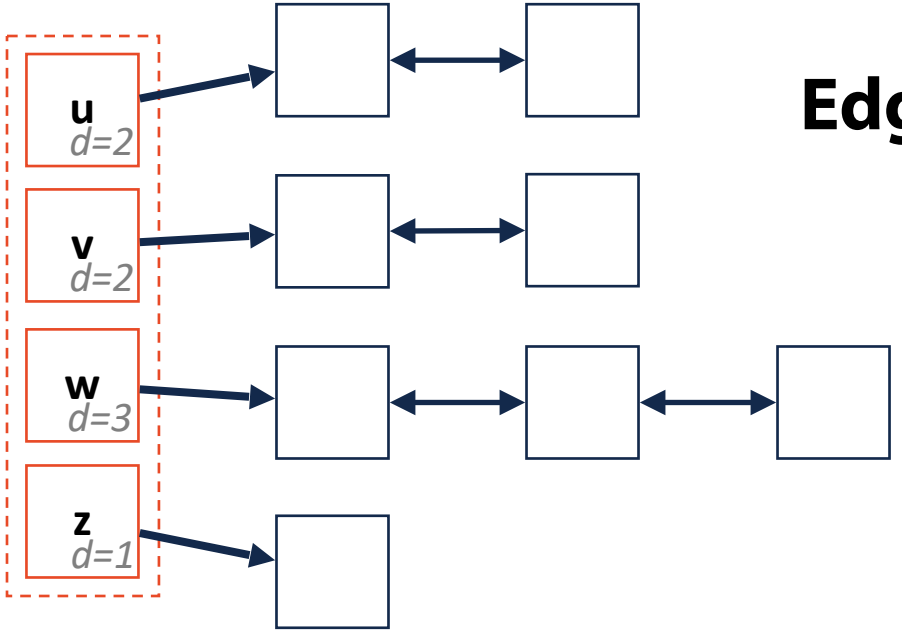
Pros:

Cons:

Adjacency List

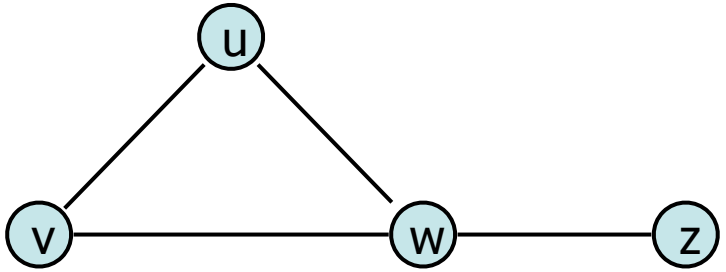


Vertex Storage:

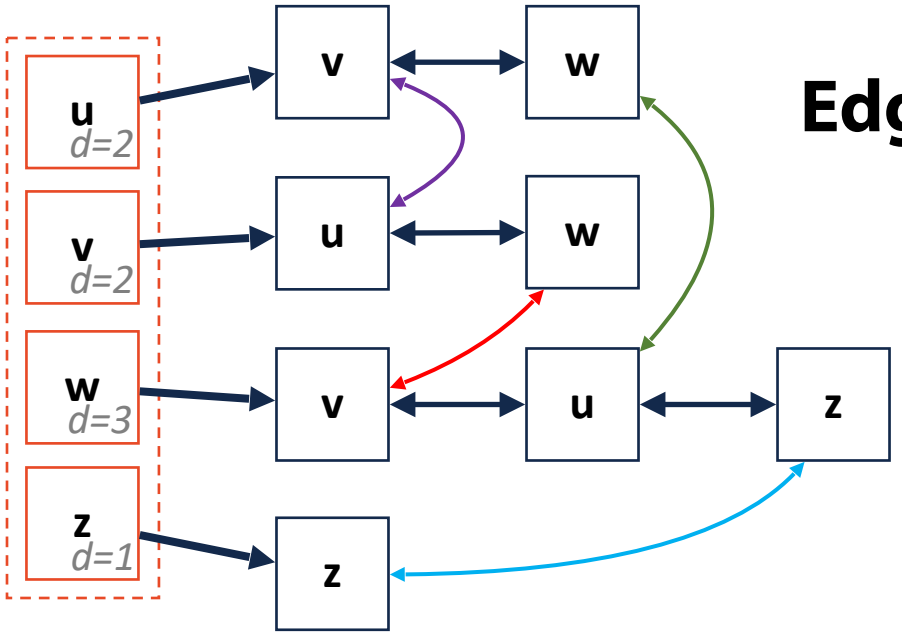


Edge Storage:

Adjacency List

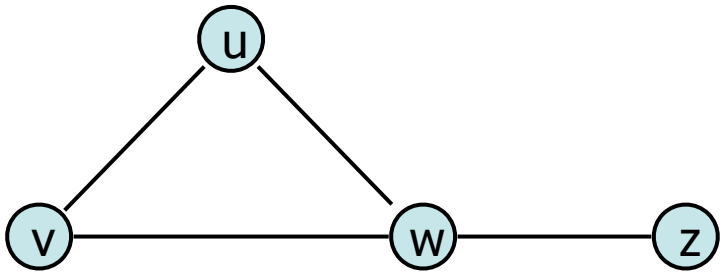


Vertex Storage:

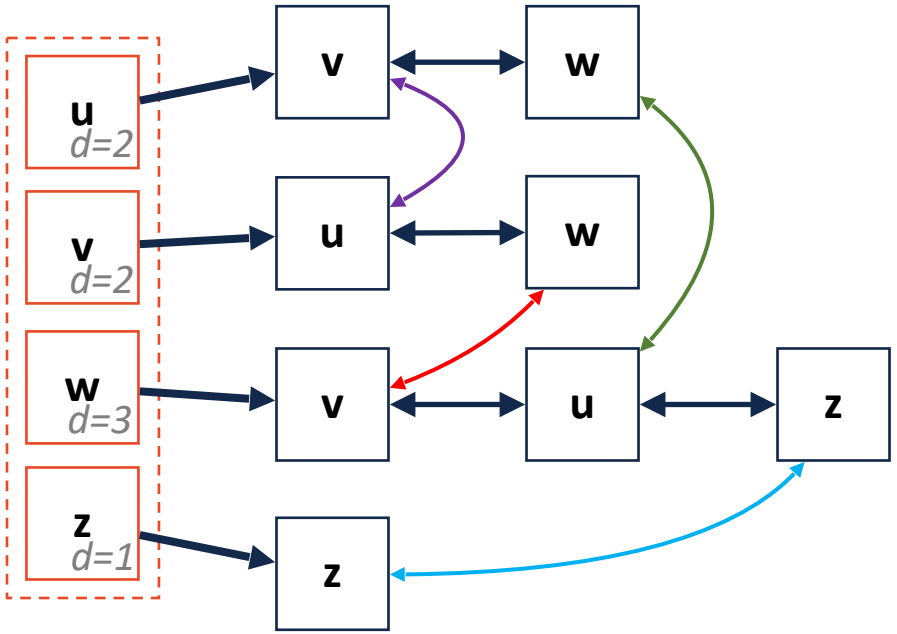


Edge Storage:

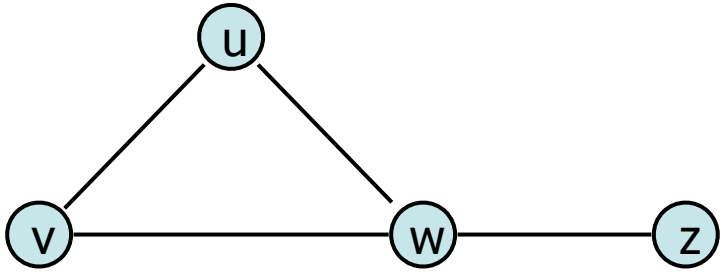
Adjacency List



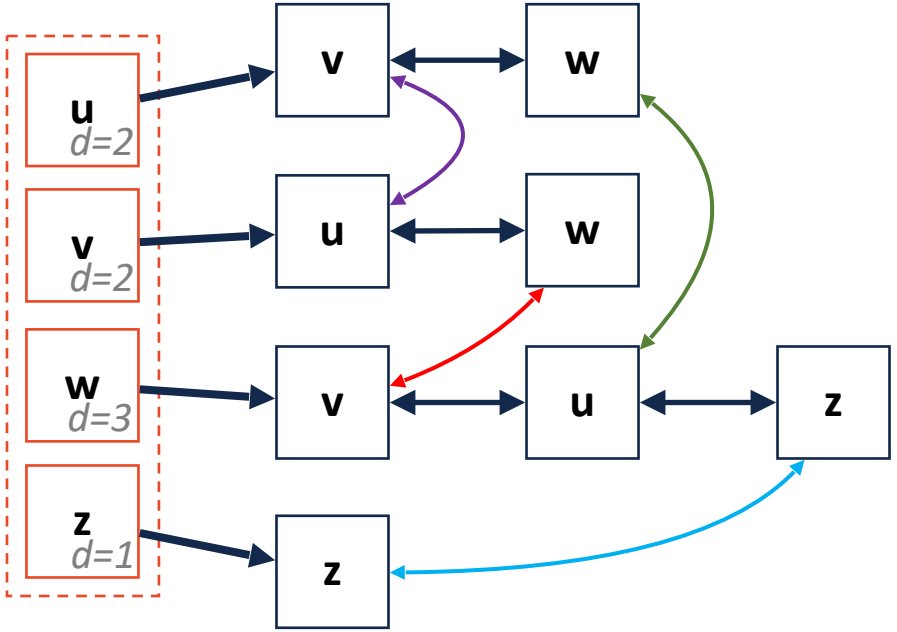
insertVertex(v):



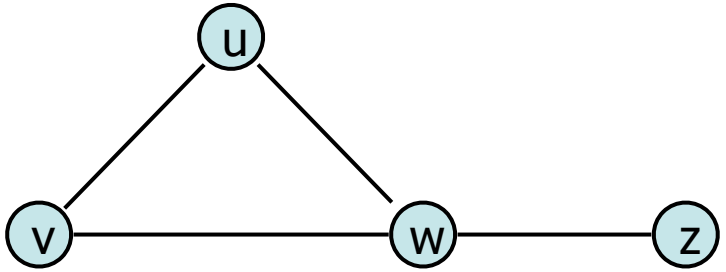
Adjacency List



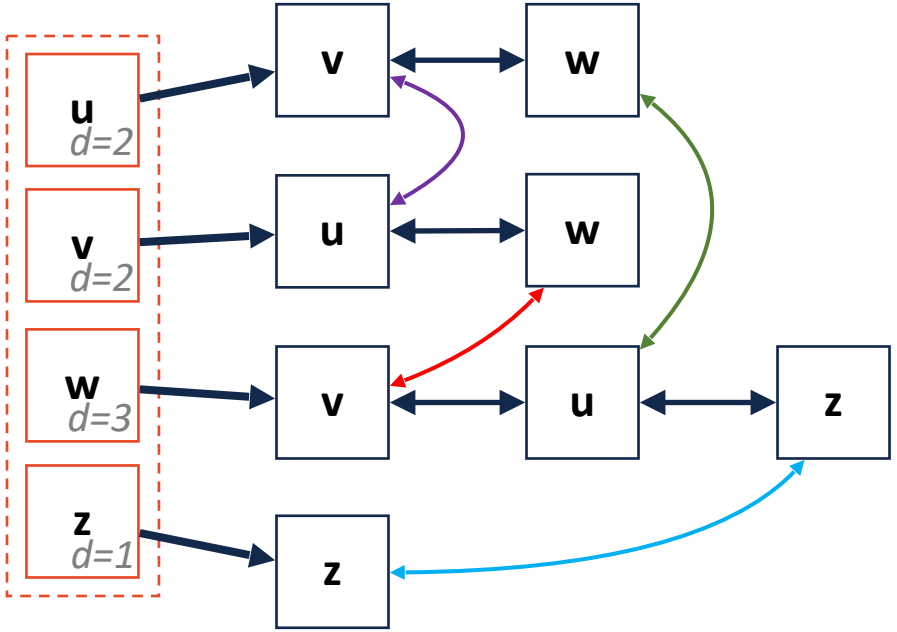
removeVertex(v):



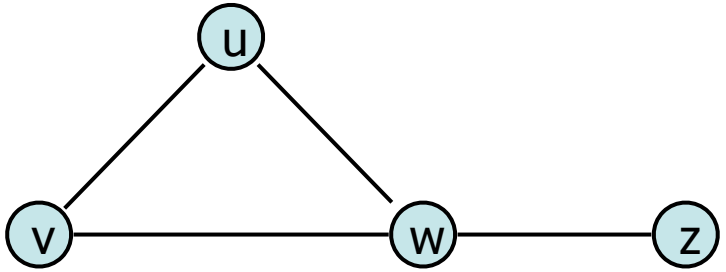
Adjacency List



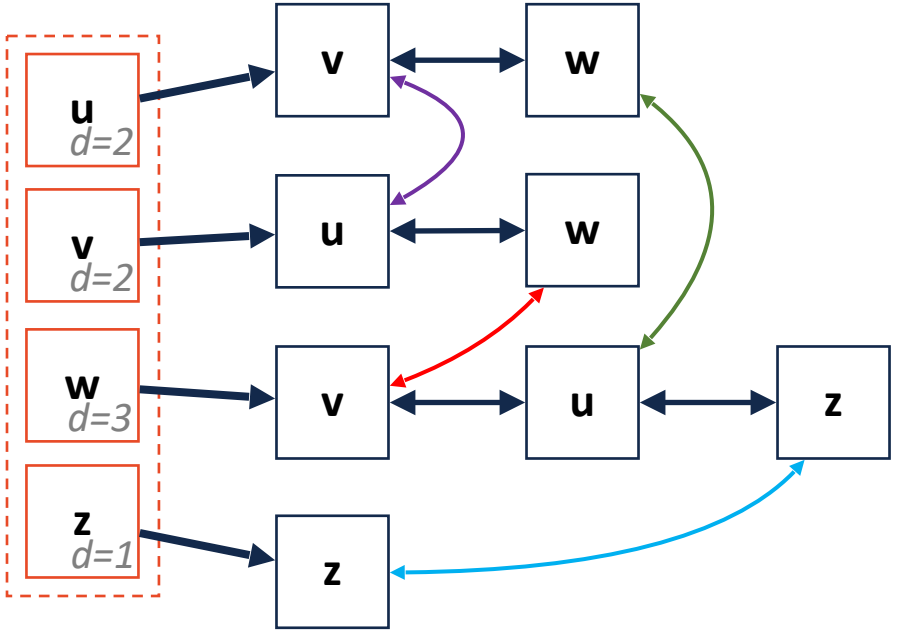
insertEdge(u, v):



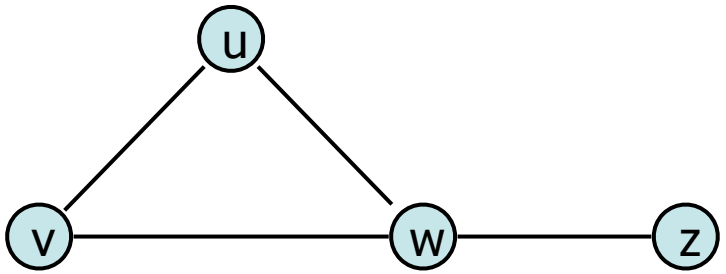
Adjacency List



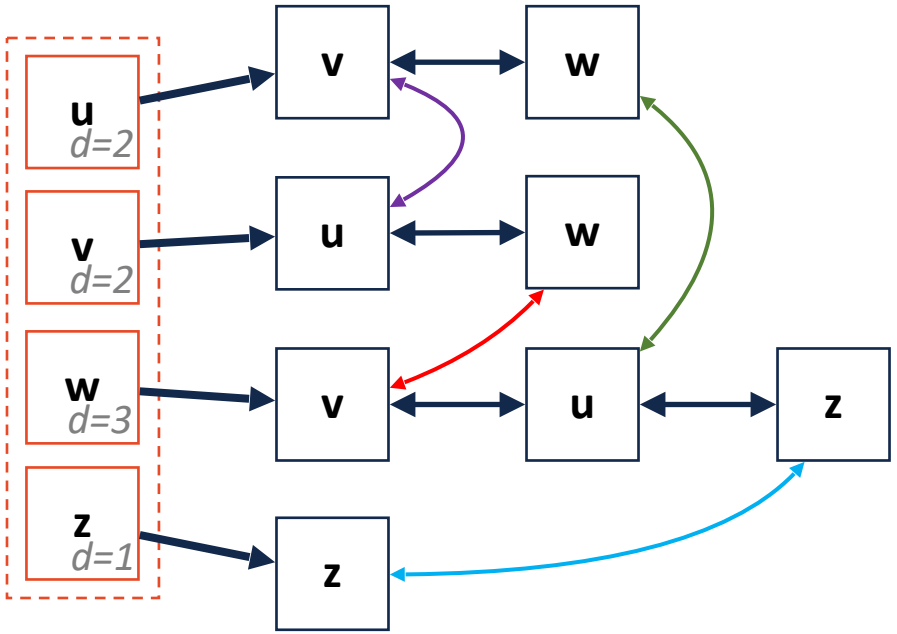
removeEdge(u, v):



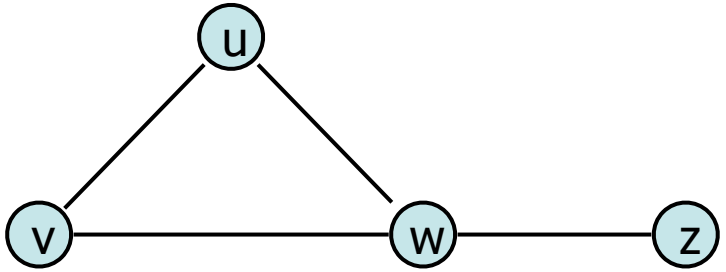
Adjacency List



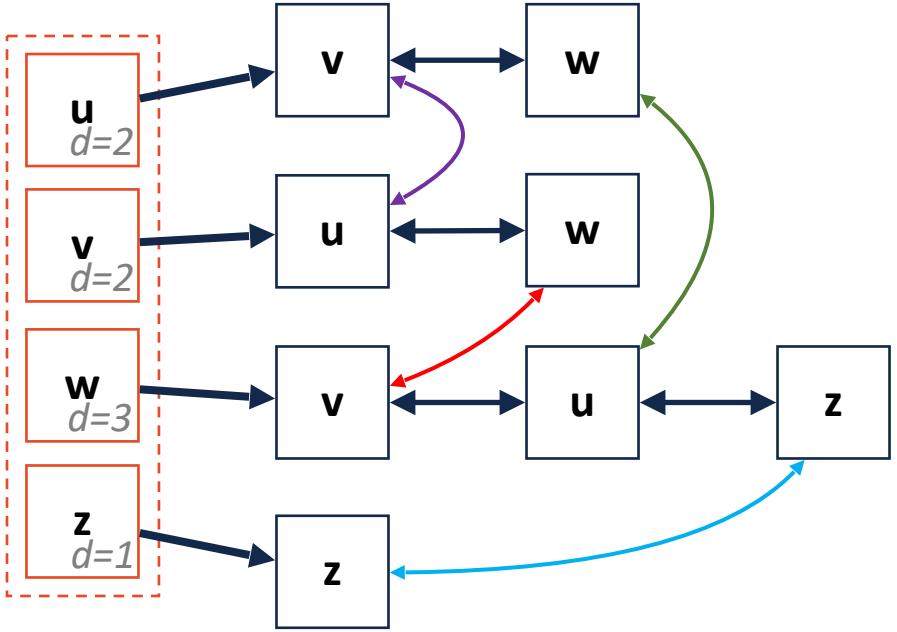
getEdges(u):



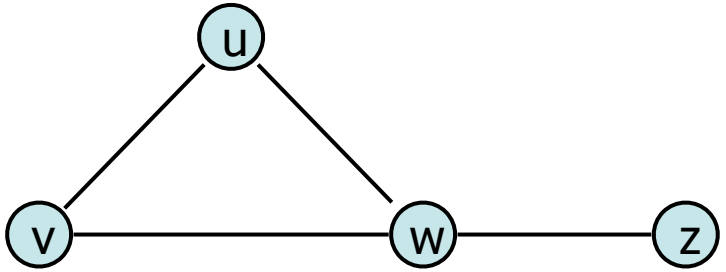
Adjacency List

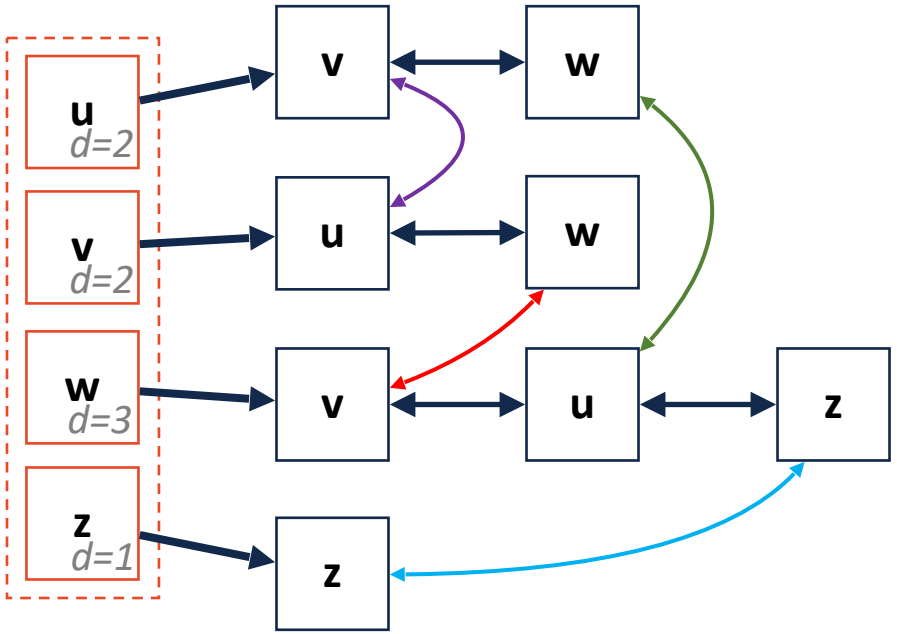


areAdjacent(u, v):



Adjacency List





Pros:

Cons:

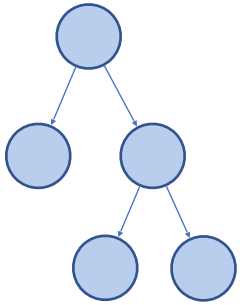
Expressed as O(f)	Edge List	Adjacency Matrix	Adjacency List
Space	n+m	n^2	n+m
insertVertex(v)	1*	n*	1*
removeVertex(v)	m**	n	deg(v)
insertEdge(u, v)	1	1	1*
removeEdge(u, v)	m	1	min(deg(u), deg(v))
getEdges(v)	m	n	deg(v)
areAdjacent(u, v)	m	1	min(deg(u), deg(v))

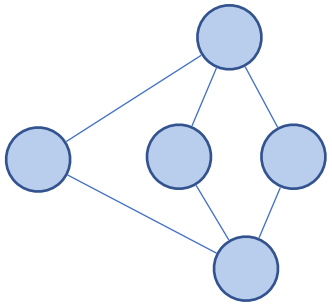
Traversal:

Objective: Visit every vertex and every edge in the graph.

Purpose: Search for interesting sub-structures in the graph.

We've seen traversal before ….but it's different:





- Ordered
- Obvious Start
-

-
-
-