

CS 225

Data Structures

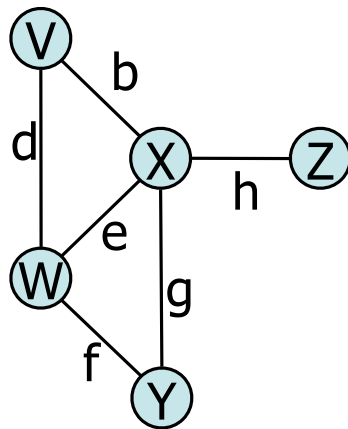
November 5 – Graph Implementation

G Carl Evans

Graph ADT

Data:

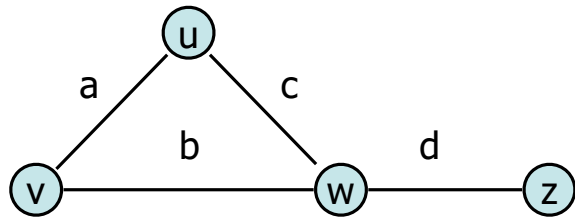
- Vertices
- Edges
- Some data structure maintaining the structure between vertices and edges.



Functions:

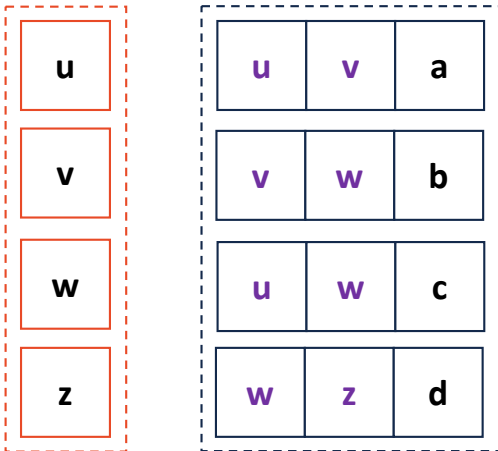
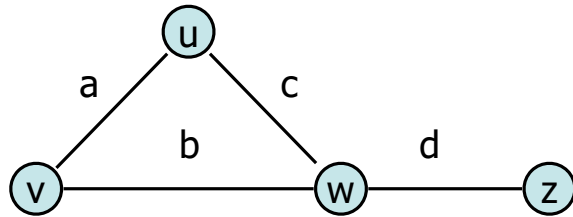
- insertVertex(K key);
- insertEdge(Vertex v1, Vertex v2, K key);
- removeVertex(Vertex v);
- removeEdge(Vertex v1, Vertex v2);
- incidentEdges(Vertex v);
- areAdjacent(Vertex v1, Vertex v2);

Graph Implementation Idea



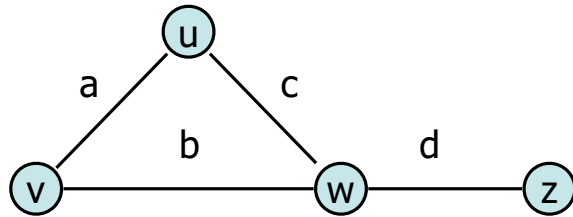
Graph Implementation: Edge List

Vertex Collection:



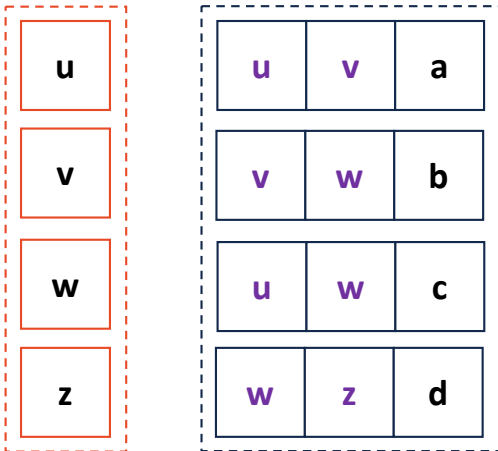
Edge Collection:

Graph Implementation: Edge List

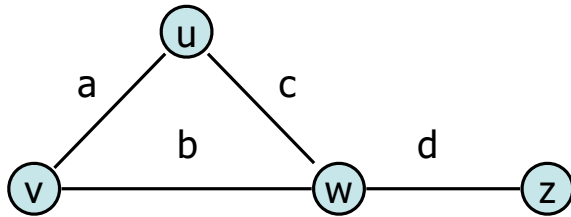


insertVertex(K key):

removeVertex(Vertex v):



Graph Implementation: Edge List



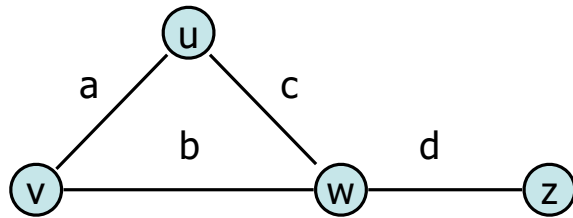
u	u	v	a
v	v	w	b
w	u	w	c
z	w	z	d

incidentEdges(Vertex v):

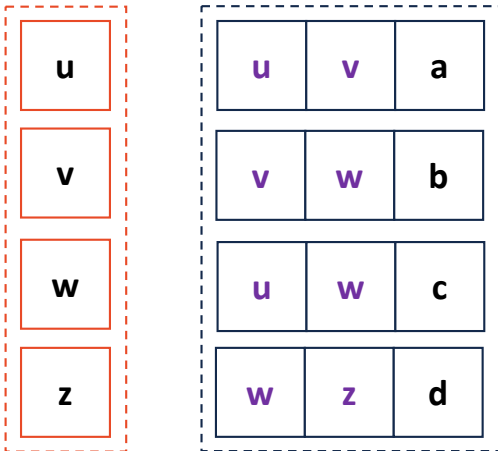
areAdjacent(Vertex v1, Vertex v2):

`G.incidentEdges(v1).contains(v2)`

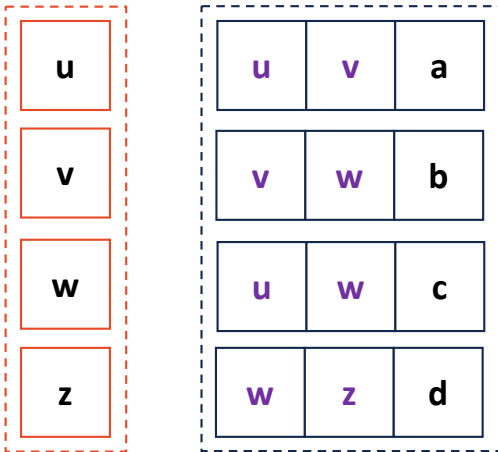
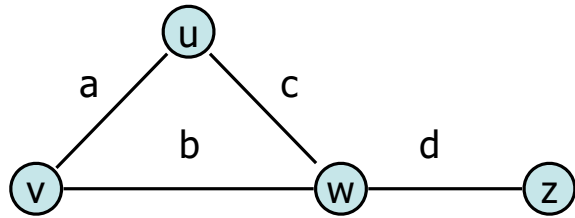
Graph Implementation: Edge List



insertEdge(Vertex v1, Vertex v2, K key):

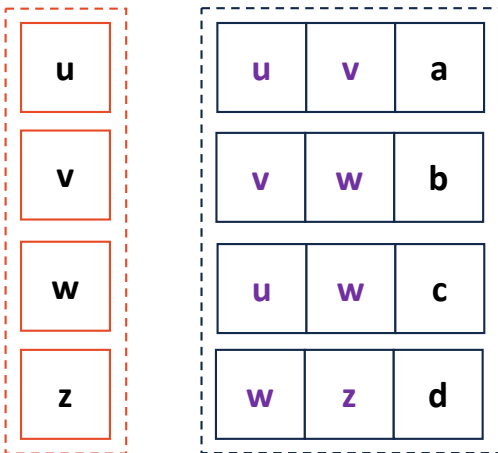
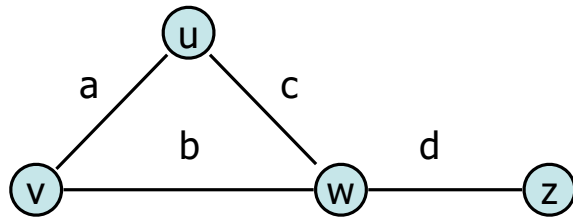


Graph Implementation: Adjacency Matrix



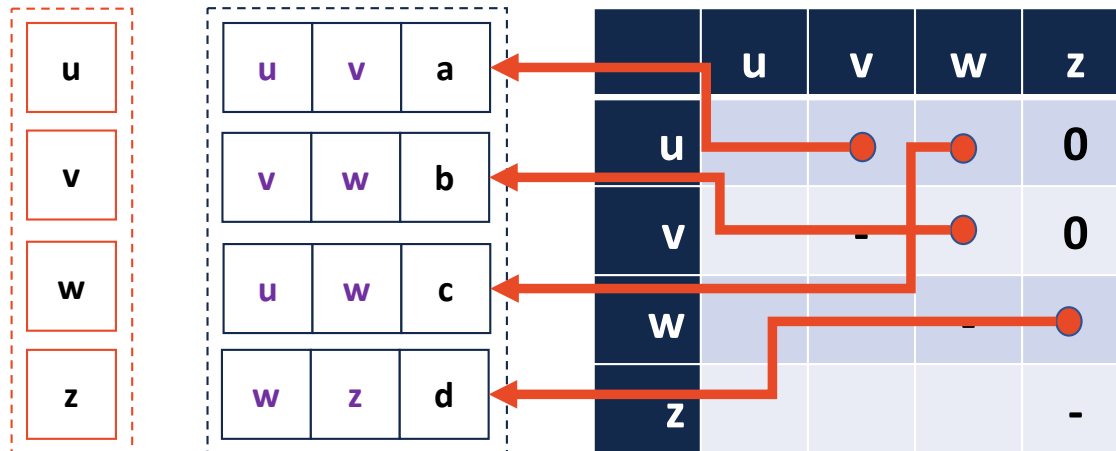
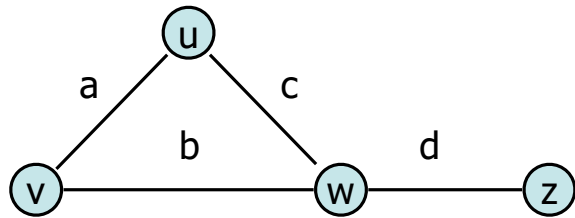
	u	v	w	z
u				
v				
w				
z				

Graph Implementation: Adjacency Matrix



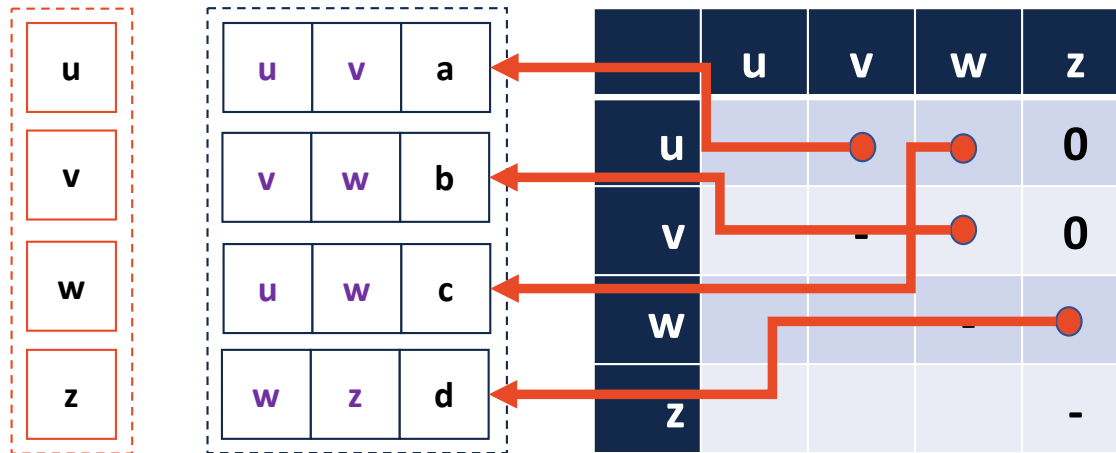
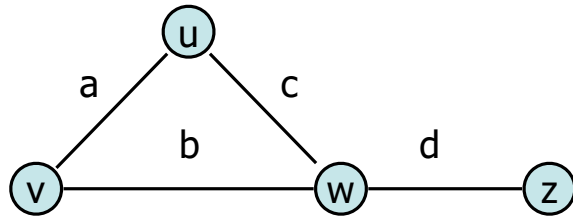
	u	v	w	z
u	-	1	1	0
v		-	1	0
w			-	1
z				-

Graph Implementation: Adjacency Matrix



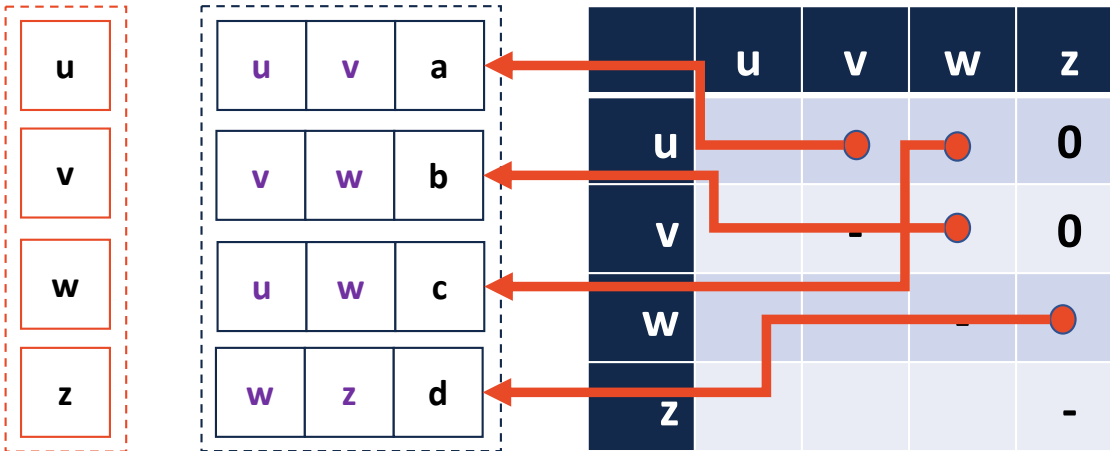
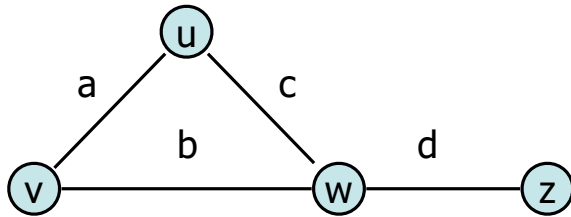
Graph Implementation: Adjacency Matrix

insertVertex(K key):



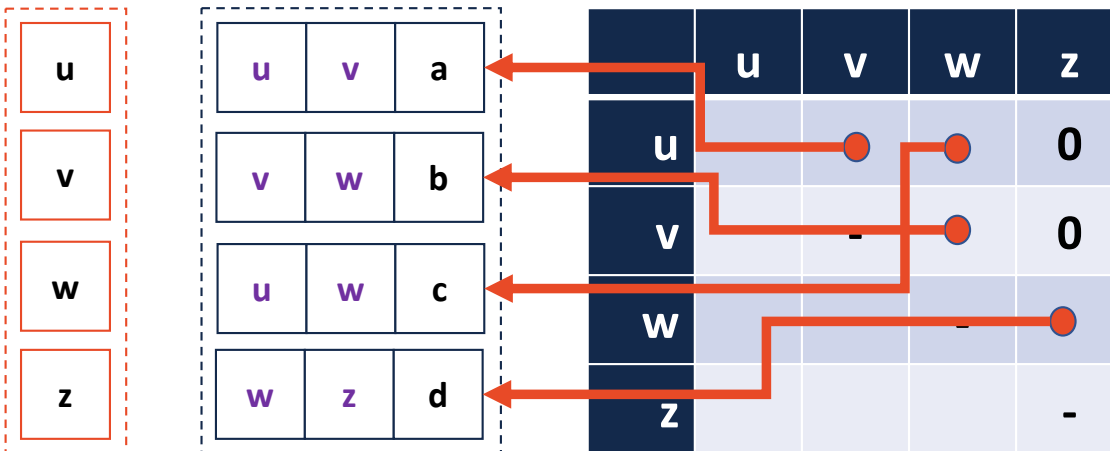
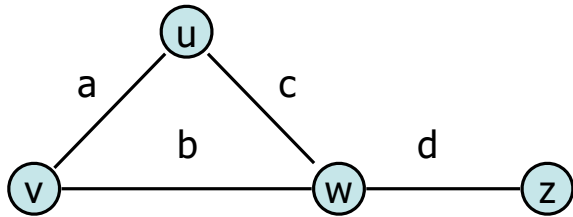
Graph Implementation: Adjacency Matrix

removeVertex(Vertex v):



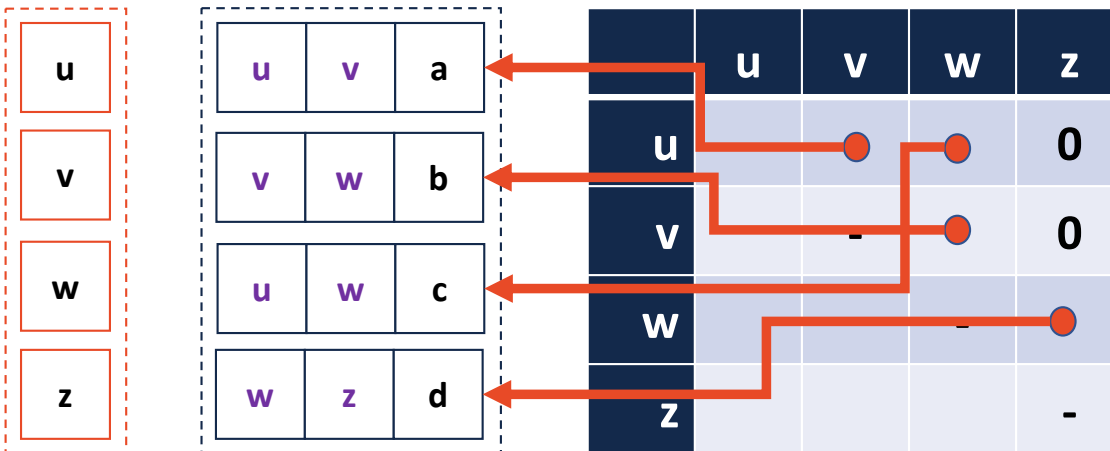
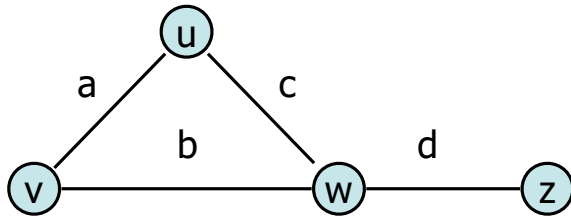
Graph Implementation: Adjacency Matrix

incidentEdges(Vertex v):



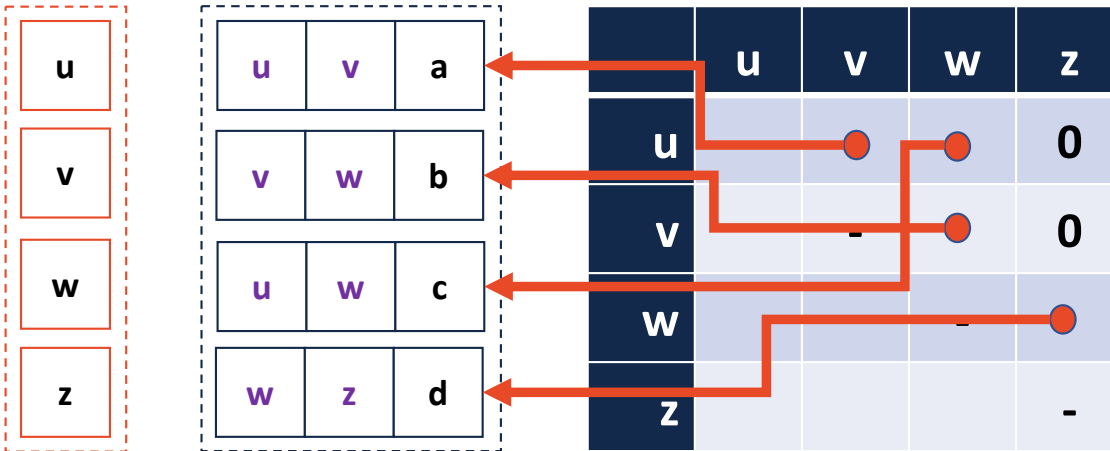
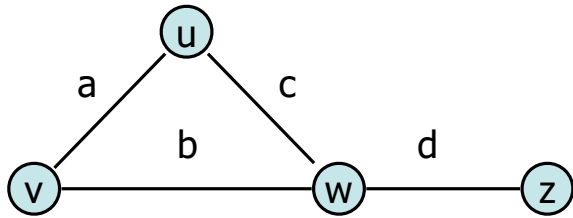
Graph Implementation: Adjacency Matrix

areAdjacent(Vertex v1, Vertex v2):

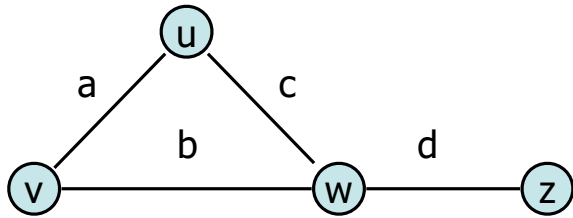


Graph Implementation: Adjacency Matrix

insertEdge(Vertex v1, Vertex v2, K key):



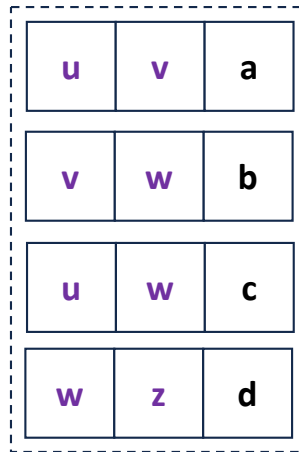
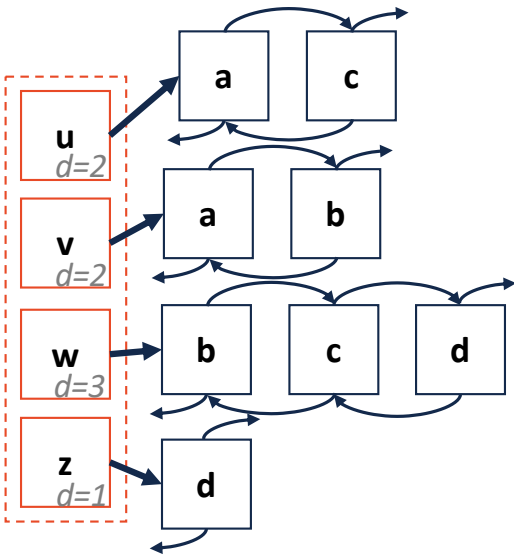
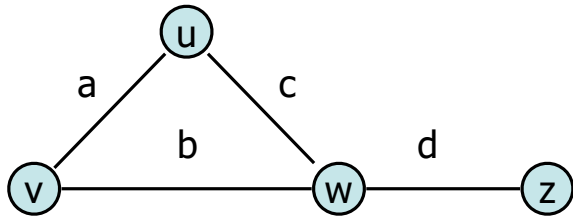
Graph Implementation: Edge List



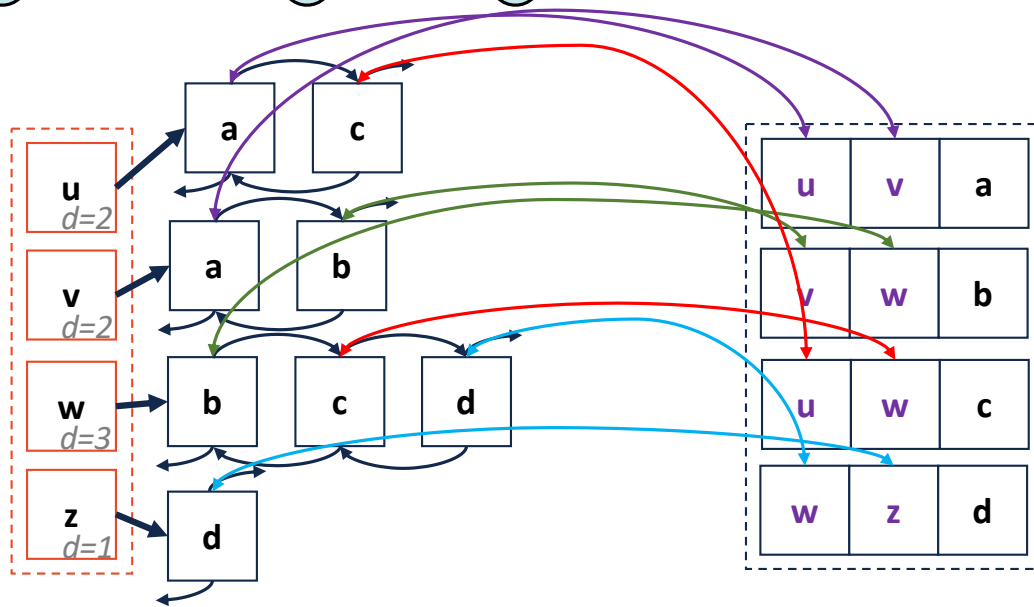
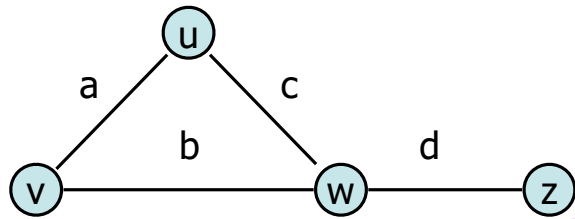
u
v
w
z

u	v	a
v	w	b
u	w	c
w	z	d

Adjacency List

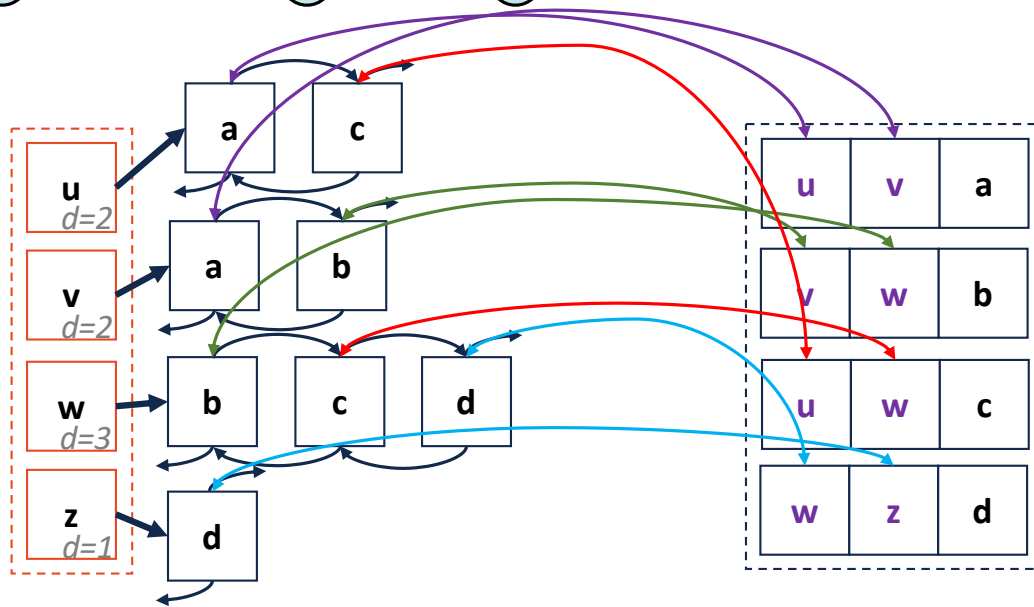
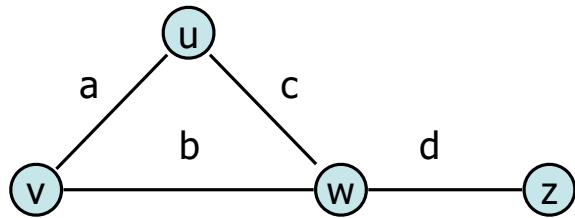


Adjacency List



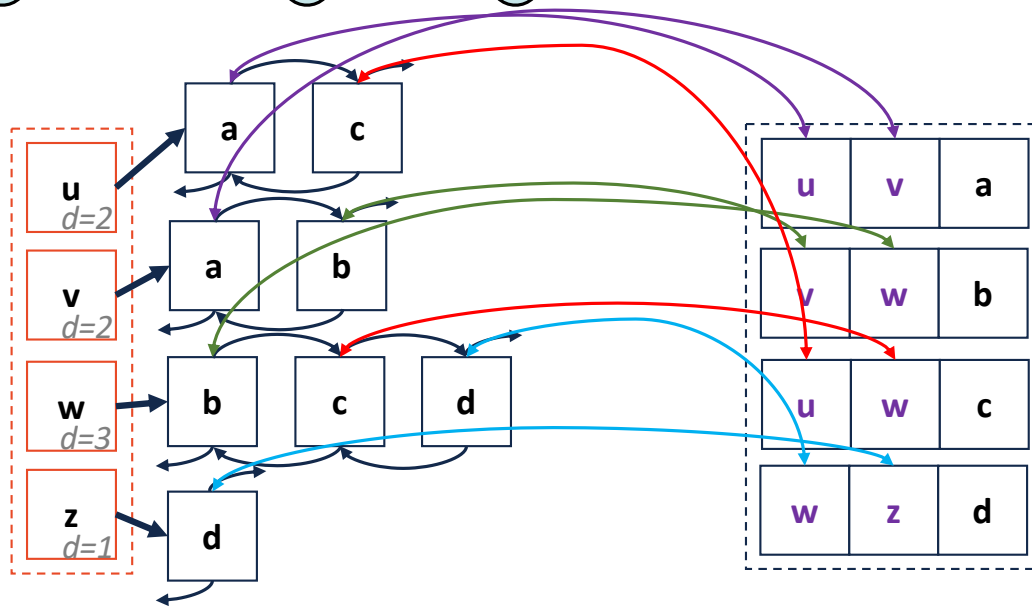
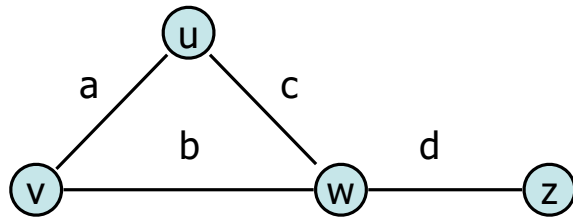
Adjacency List

insertVertex(K key):



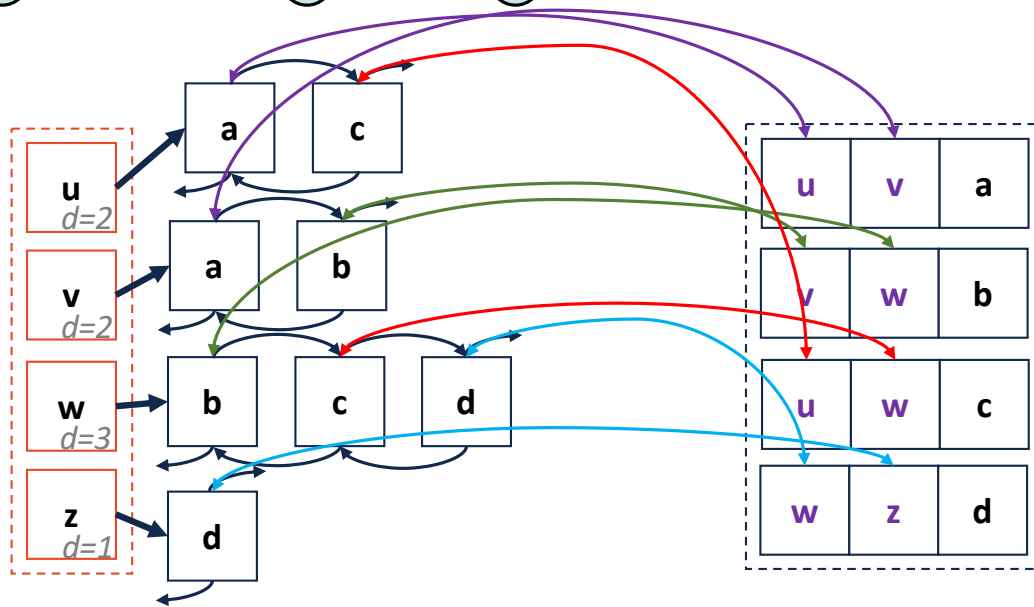
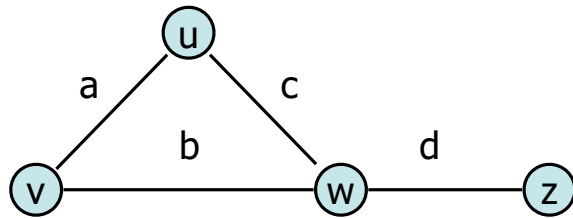
Adjacency List

removeVertex(Vertex v):



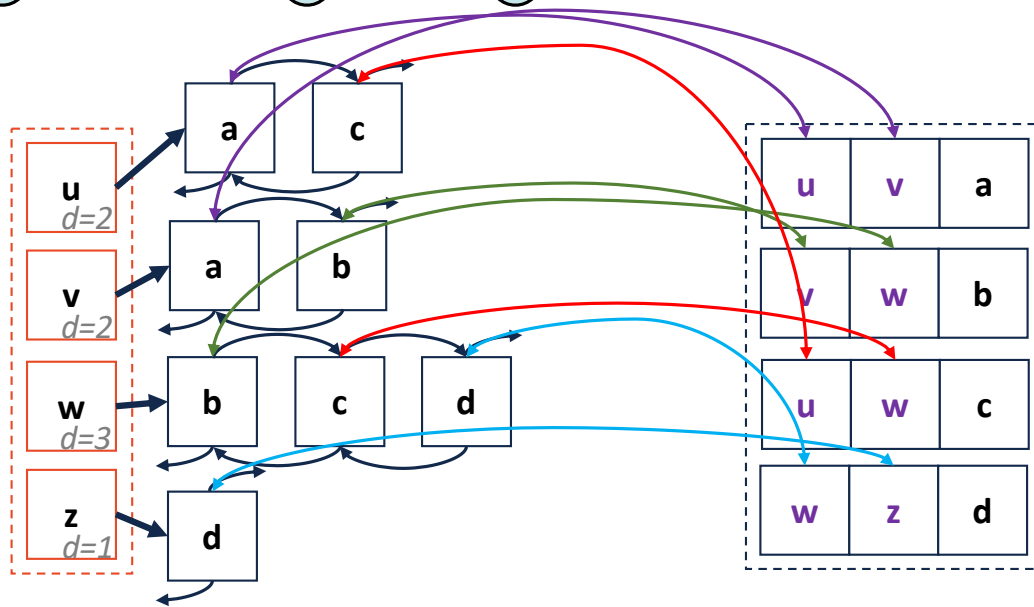
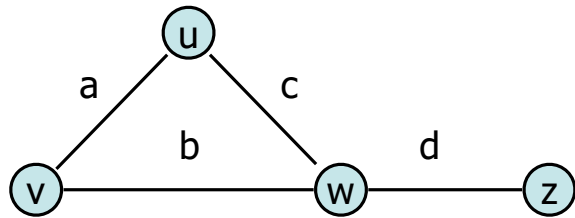
Adjacency List

incidentEdges(Vertex v):



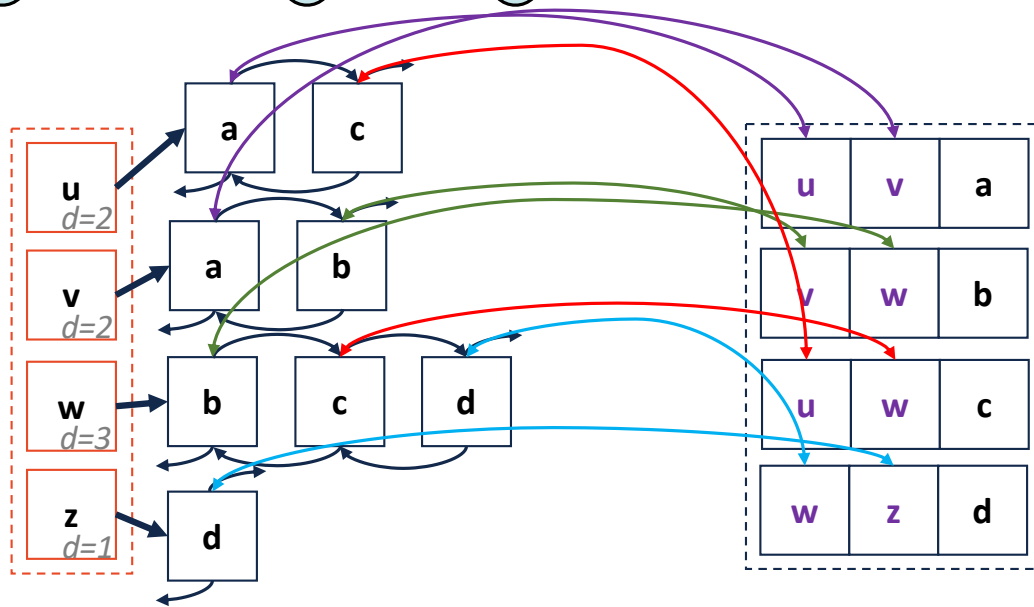
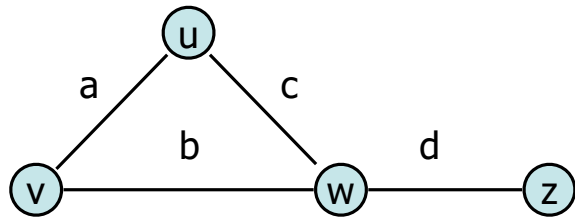
Adjacency List

areAdjacent(Vertex v1, Vertex v2):



Adjacency List

insertEdge(Vertex v1, Vertex v2, K key):



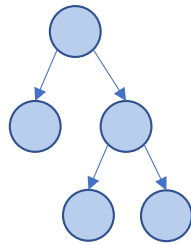
Expressed as O(f)	Edge List	Adjacency Matrix	Adjacency List
Space	$n+m$	n^2	$n+m$
insertVertex(v)	1	n	1
removeVertex(v)	m	n	deg(v)
insertEdge(v, w, k)	1	1	1
removeEdge(v, w)	1	1	1
incidentEdges(v)	m	n	deg(v)
areAdjacent(v, w)	m	1	min(deg(v), deg(w))

Traversal:

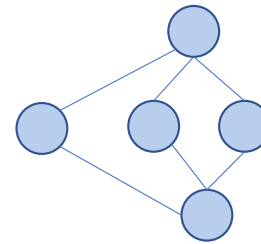
Objective: Visit every vertex and every edge in the graph.

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

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



- Ordered
- Obvious Start
-



-
-
-

Traversal: BFS

