Data Structures

Graph Implementations

CS 225

November 13, 2023

Brad Solomon & G Carl Evans

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

MP_Sketching Releases Today

A brand new assignment

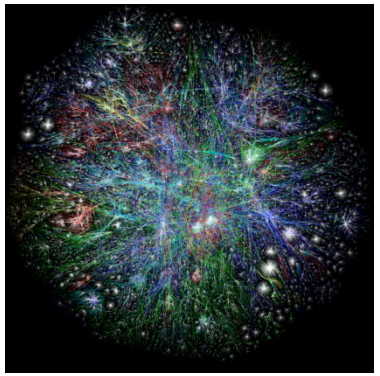
If instructions are unclear ask*!

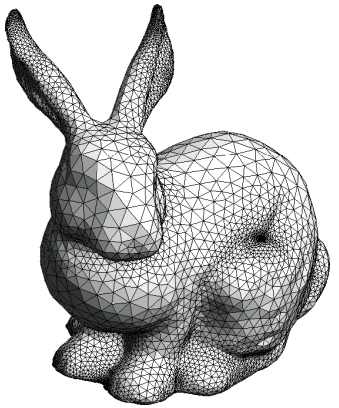
* Most staff probably won't know this assignment until Wednesday

Learning Objectives

Discuss graph implementation and storage strategies

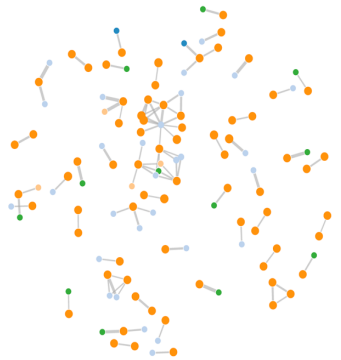
Graphs





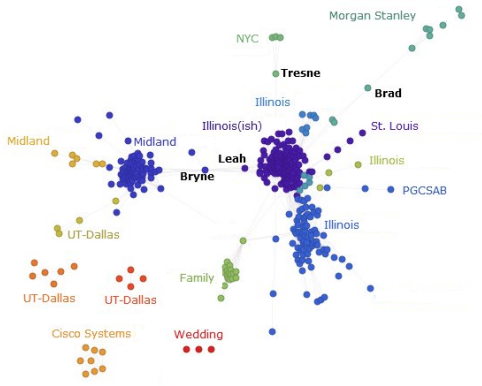
To study all of these structures:

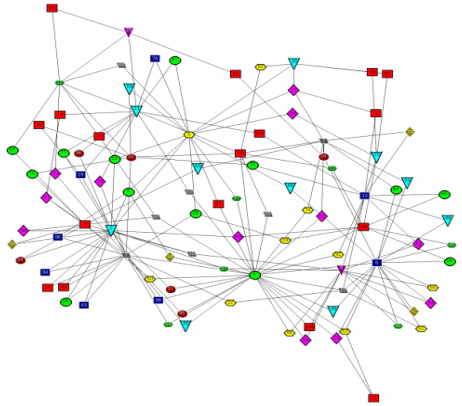
1. A common vocabulary
2. Graph implementations
3. Graph traversals
4. Graph algorithms

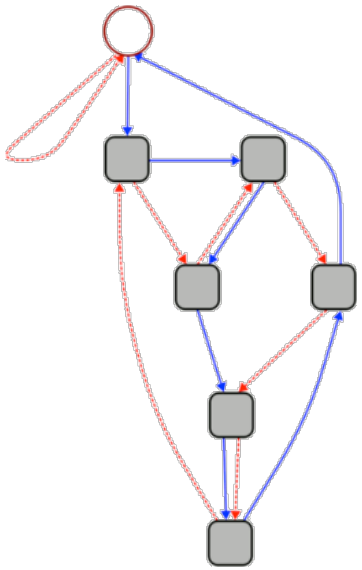


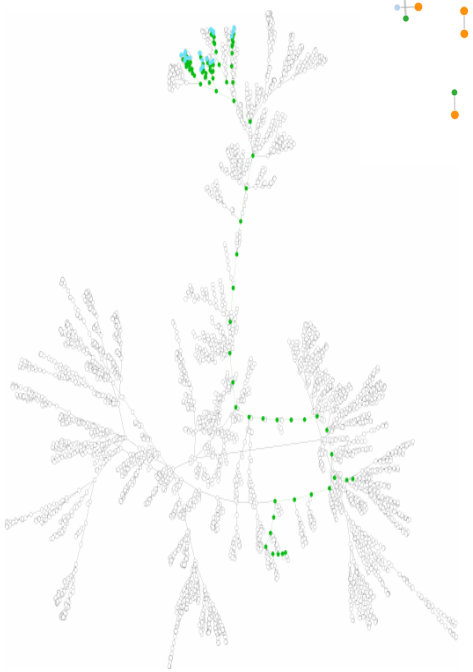
HAMLET

TROILUS AND CRESSIDA





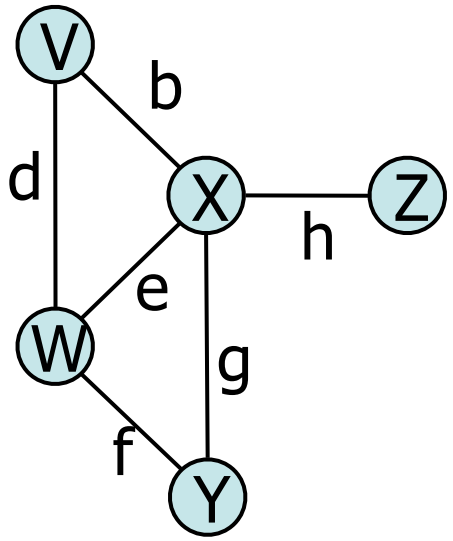




Graph ADT

Data:

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



Functions:

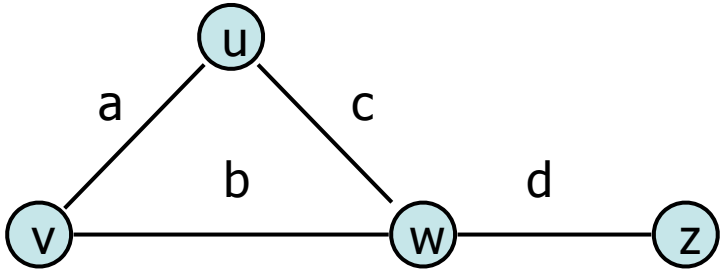
- insertVertex(K key);
- insertEdge(Vertex v1, Vertex v2, K key);

- removeVertex(Vertex v);
- removeEdge(Vertex v1, Vertex v2);

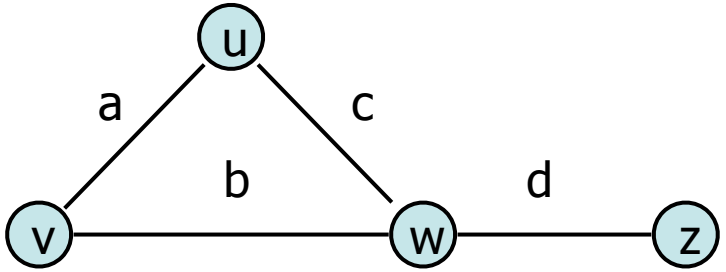
- incidentEdges(Vertex v);
- areAdjacent(Vertex v1, Vertex v2);

- origin(Edge e);
- destination(Edge e);

Graph Implementation Ideas



Graph Implementation: Edge List |V|= n, |E|= m



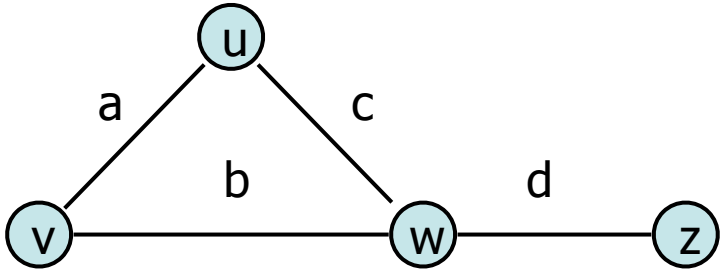
Vertex Storage:

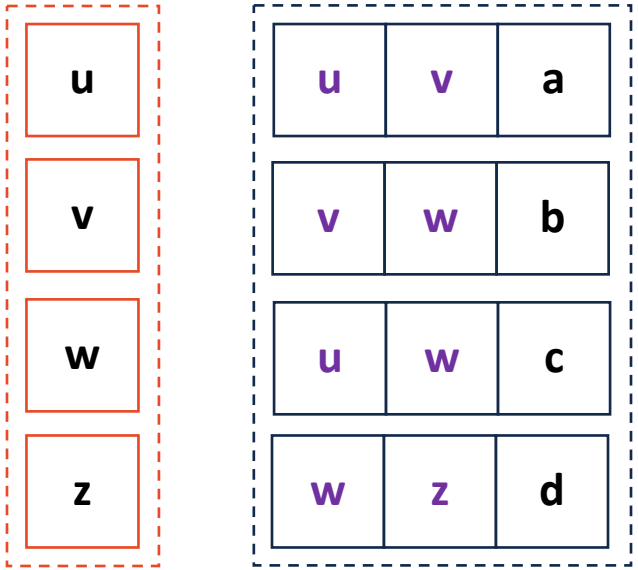
u
v
w
z

Edge Storage:

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

Graph Implementation: Edge List |V|= n, |E|= m

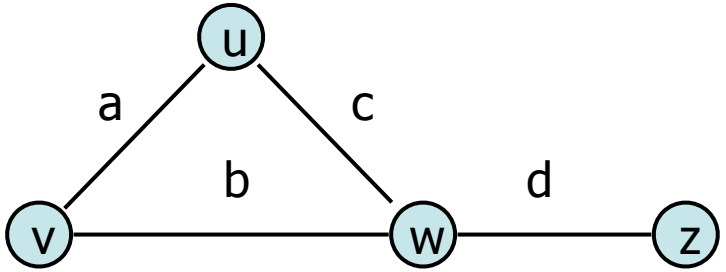




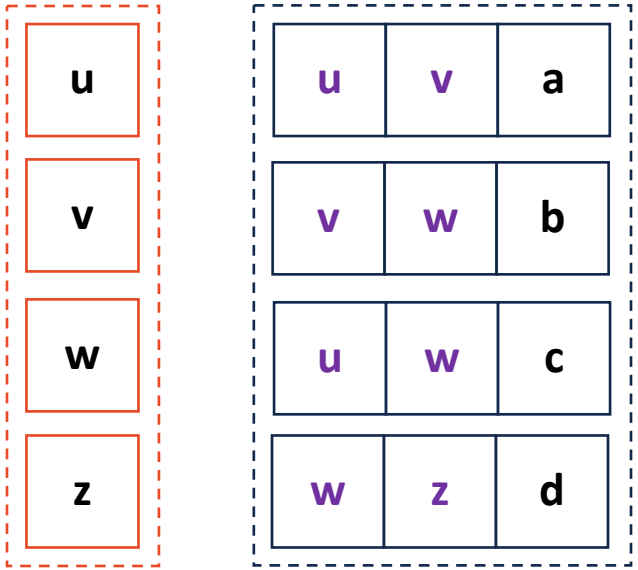
insertVertex(K key):

removeVertex(Vertex v):

Graph Implementation: Edge List |V|= n, |E|= m

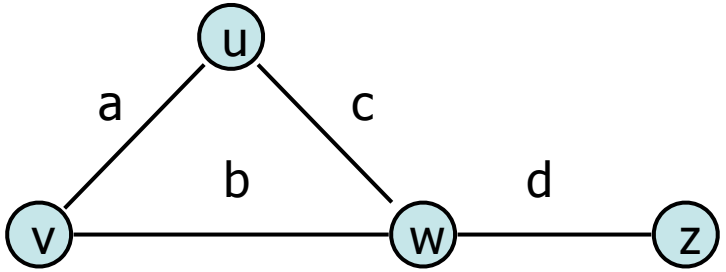


incidentEdges(Vertex v):

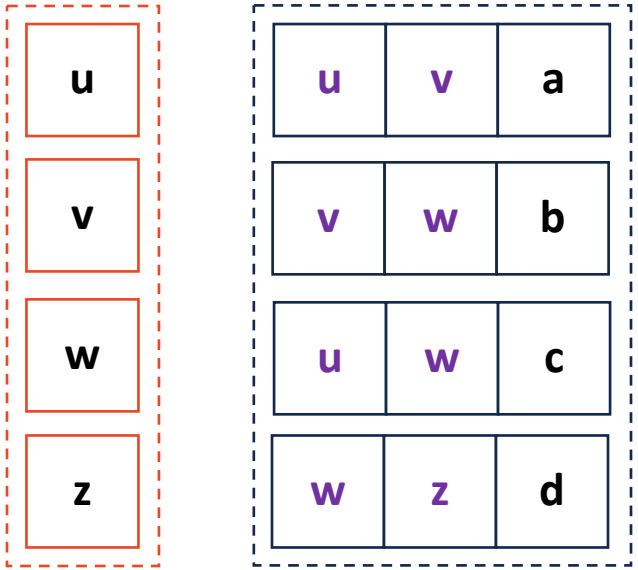


areAdjacent(Vertex v1, Vertex v2):

Graph Implementation: Edge List |V|= n, |E|= m



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



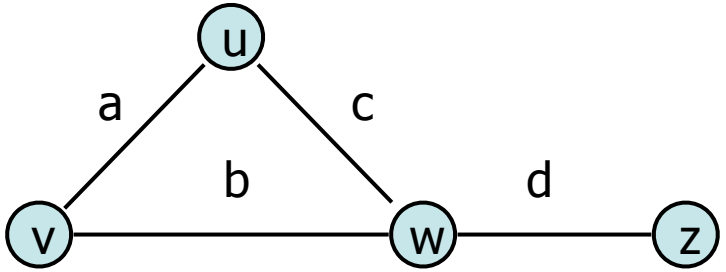
removeEdge(Vertex v1, Vertex v2, K key):

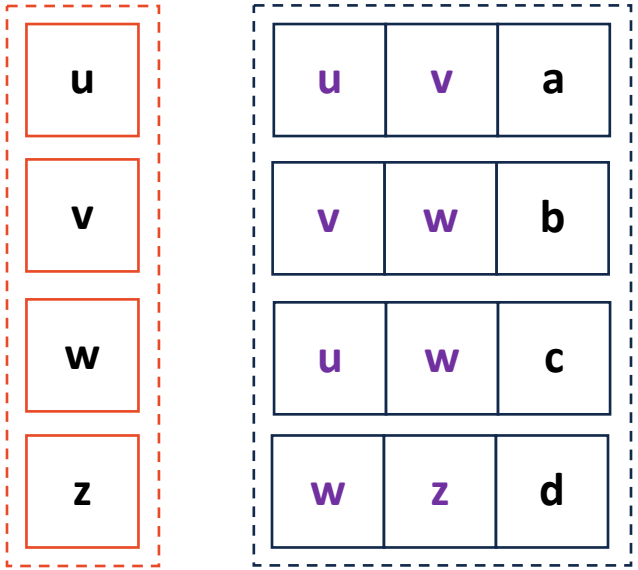
Graph Implementation: Edge List

Pros:

Cons:

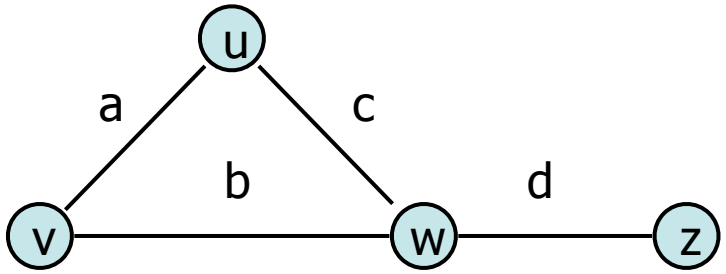
Graph Implementation: Adjacency Matrix





	u	v	w	z
u				
v				
w				
z				

Graph Implementation: Adjacency Matrix

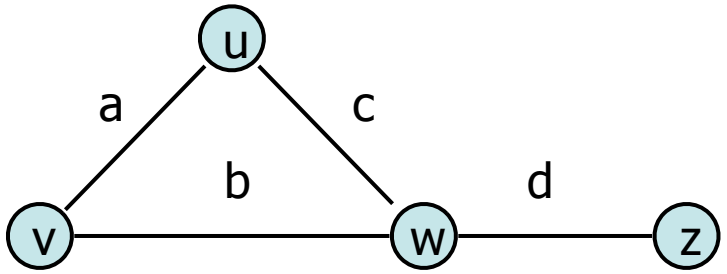


u	0
v	1
w	2
z	3

	0	1	2	3
0				
1				
2				
3				

Graph Implementation: Adjacency Matrix

|V|= n,|E|= m



Vertex Storage:

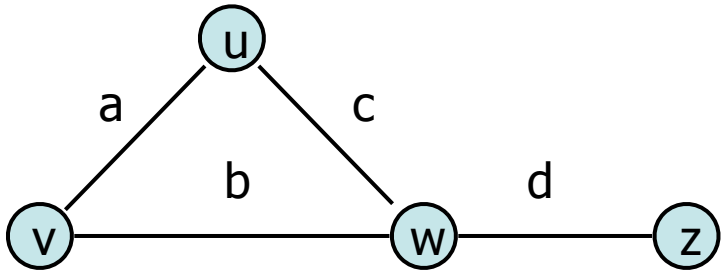
u	0
v	1
w	2
z	3

	0	1	2	3
0	-	a	c	0
1		-	b	0
2			-	d
3				-

Edge Storage:

Alternative Adjacency Matrix Implementation

|V|= n,|E|= m



Vertex Storage:

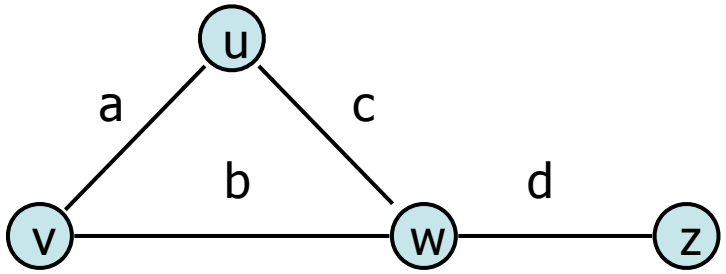
	u	v	w	z
u	-	a	c	0
v		-	b	0
w			-	d
z				-

Edge Storage:

We save ~**O(n)** space but at what cost?

Graph Implementation: Adjacency Matrix

|V|= n,|E|= m



u	0
v	1
w	2
z	3

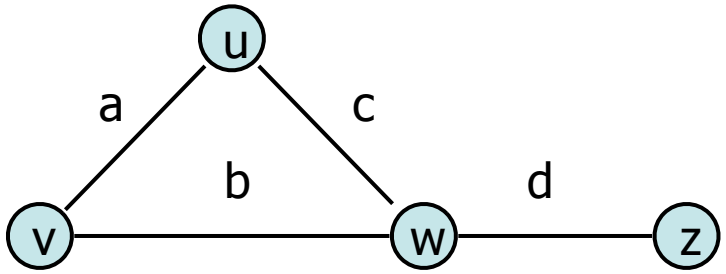
	0	1	2	3
0	-	a	c	0
1		-	b	0
2			-	d
3				-

insertVertex(K key):

removeVertex(Vertex v):

Graph Implementation: Adjacency Matrix

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



incidentEdges(Vertex v):

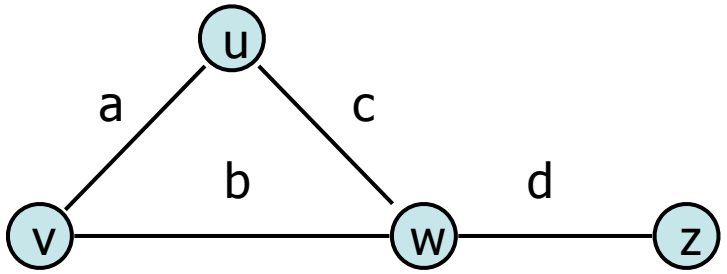
u	0
v	1
w	2
z	3

	0	1	2	3
0	-	a	c	0
1		-	b	0
2			-	d
3				-

areAdjacent(Vertex v1, Vertex v2):

Graph Implementation: Adjacency Matrix

|V|= n, |E|= m



u	0
v	1
w	2
z	3

	0	1	2	3
0	-	a	c	0
1		-	b	0
2			-	d
3				-

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

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

Graph Implementation: Adjacency Matrix

Pros:

Cons:

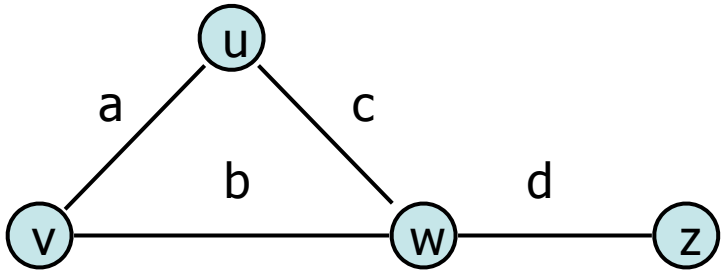
Graph Implementations

We want something…

Faster than an edge list

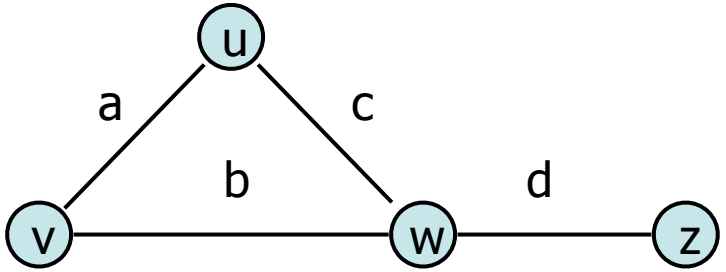
Less space than an adjacency matrix

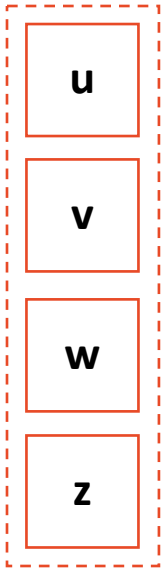
Particularly good at finding adjacent elements

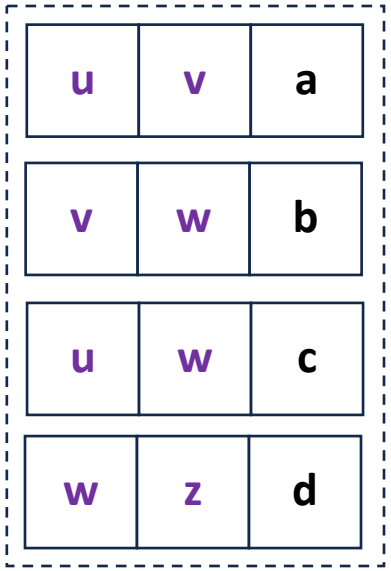


Graph Implementation: Edge List + ?

|V|= n,|E|= m

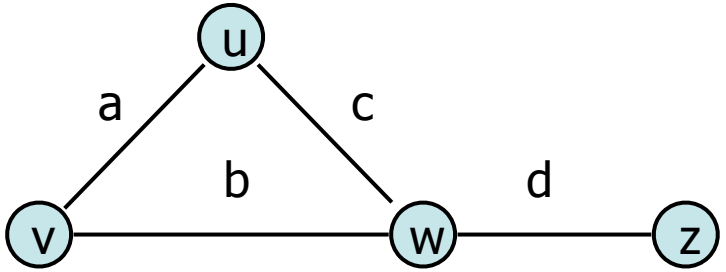


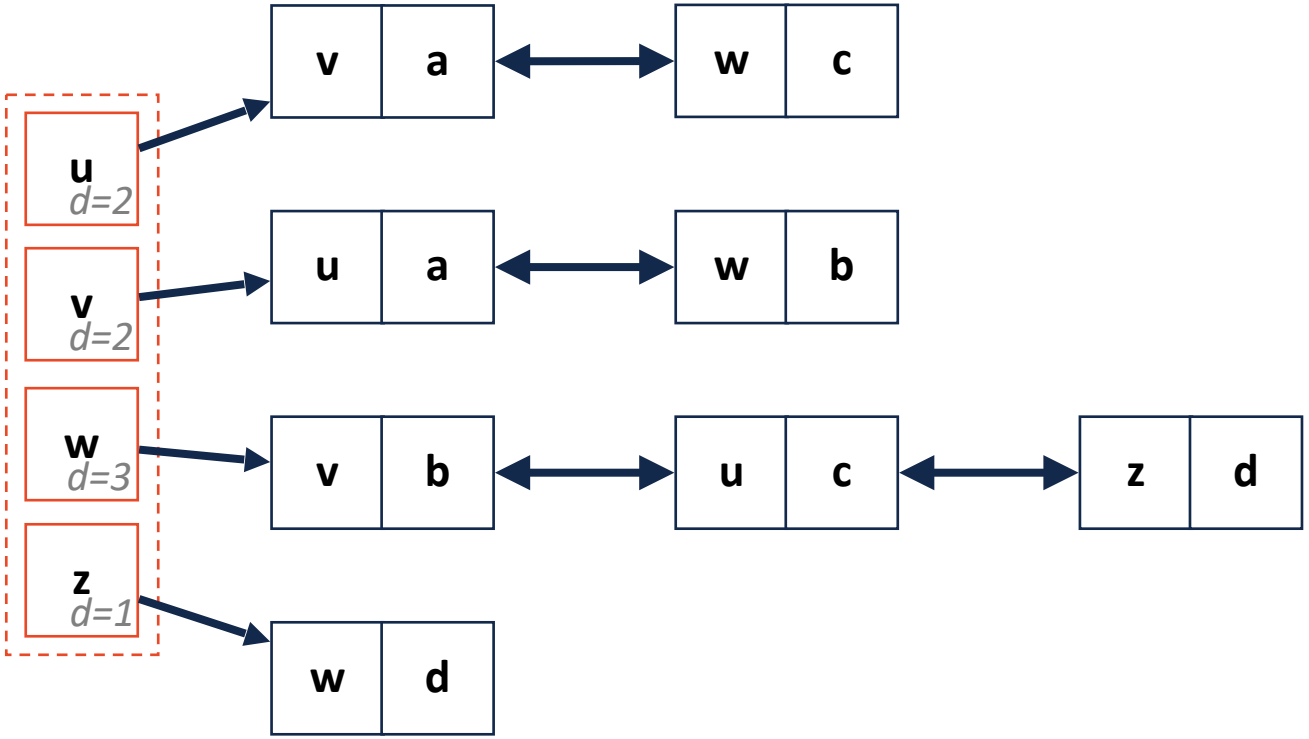




Graph Implementation: Adjacency List

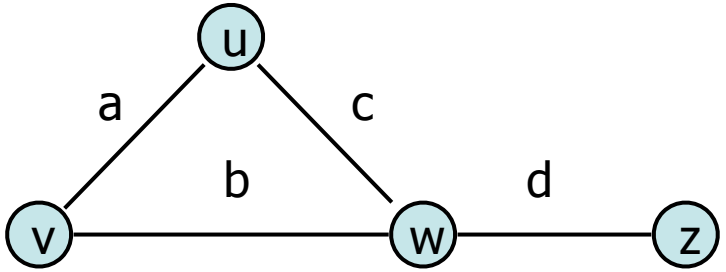
|V|= n, |E|= m



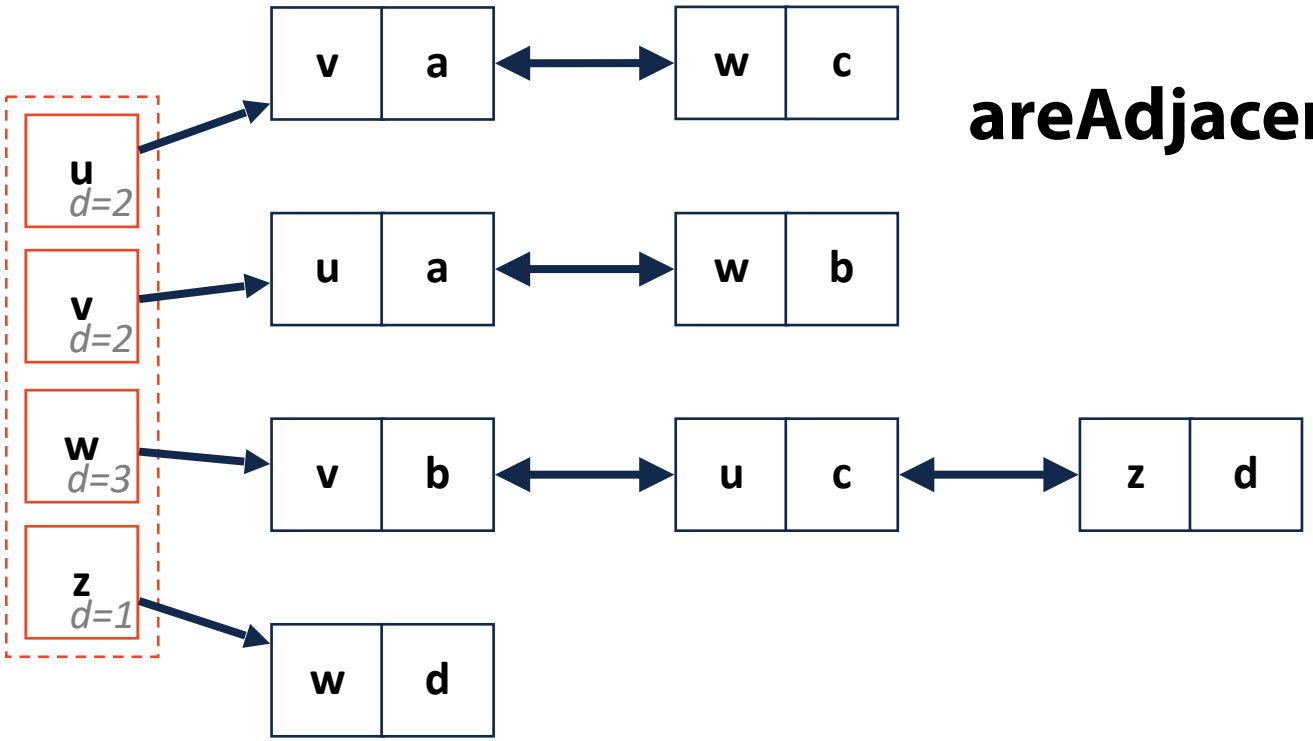


Graph Implementation: Adjacency List

|V|= n, |E|= m



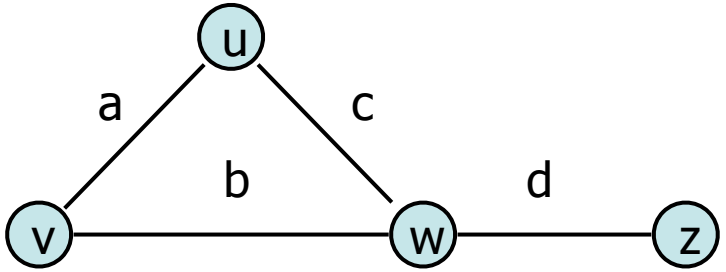
incidentEdges(Vertex v):



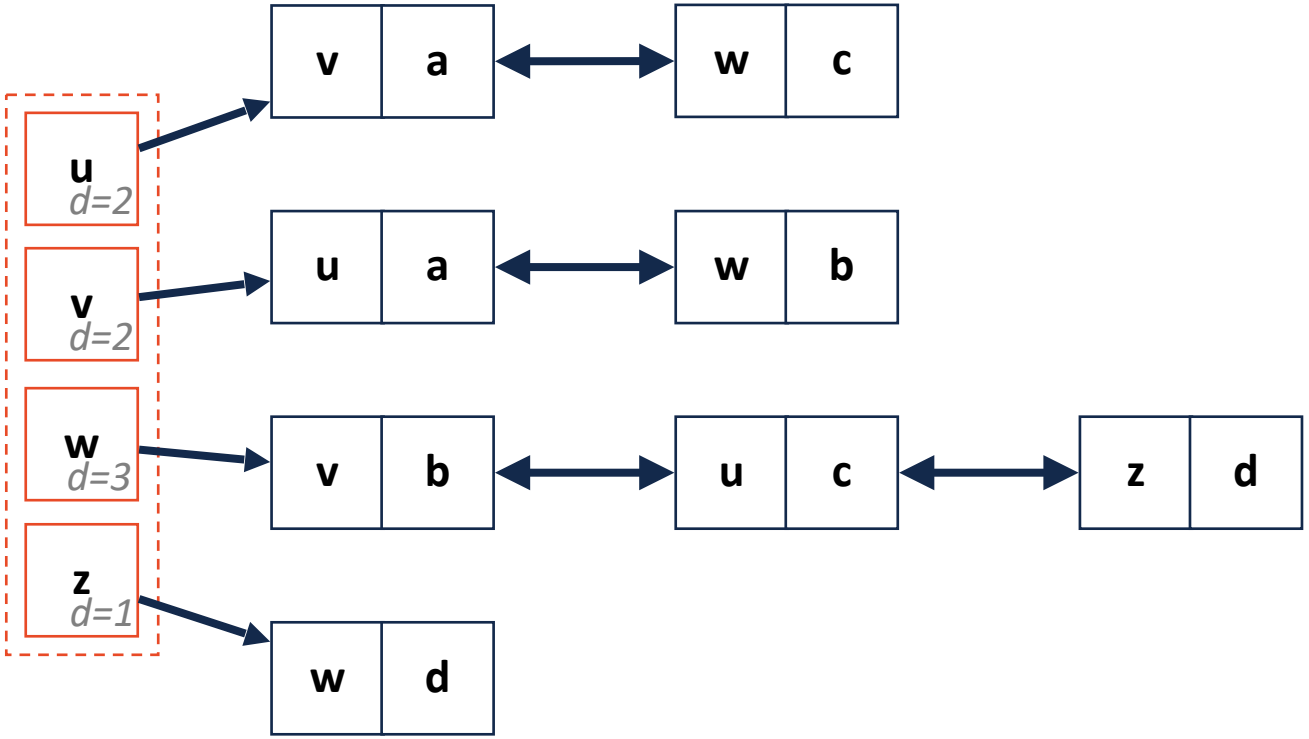
areAdjacent(Vertex v1, Vertex v2):

Graph Implementation: Adjacency List

|V|= n, |E|= m



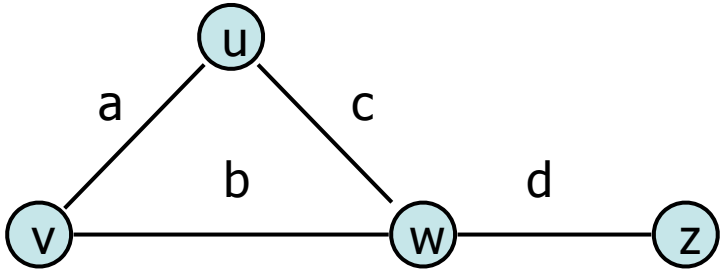
removeEdge(Vertex v1, Vertex v2, K key):

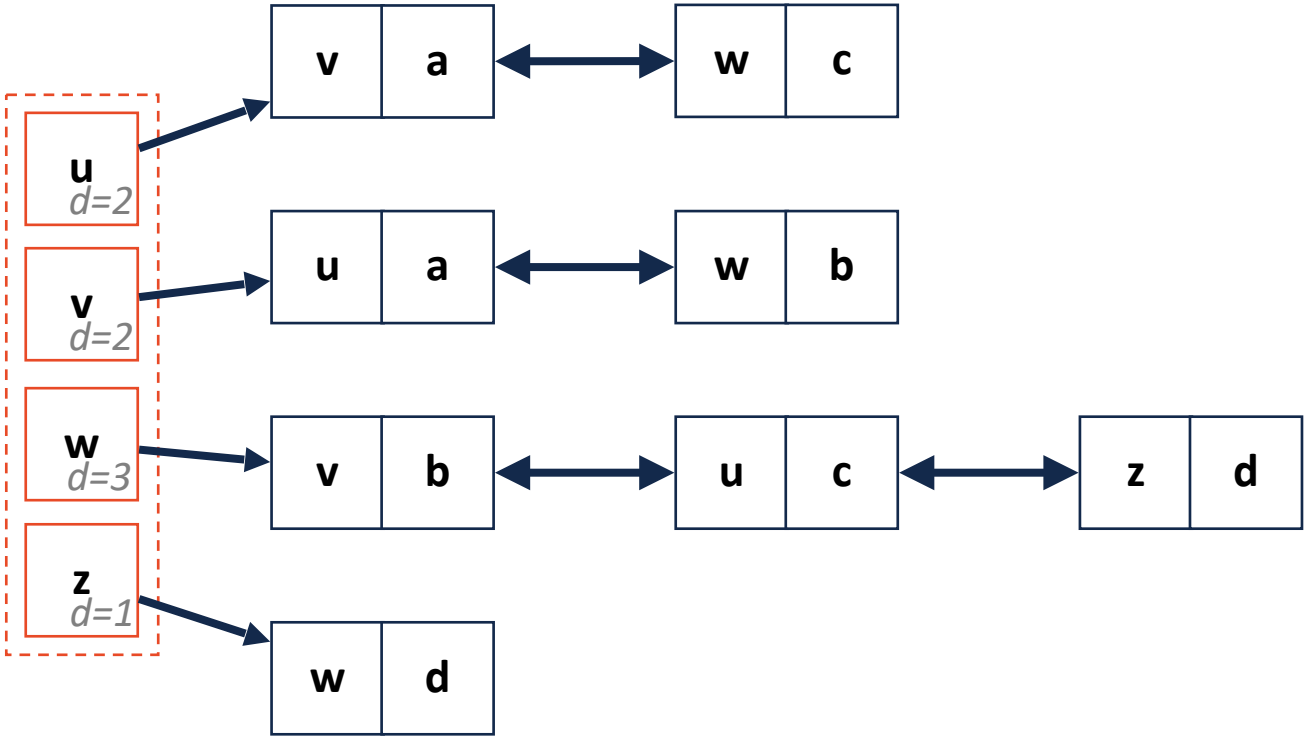


Graph Implementation: Adjacency List

|V|= n, |E|= m

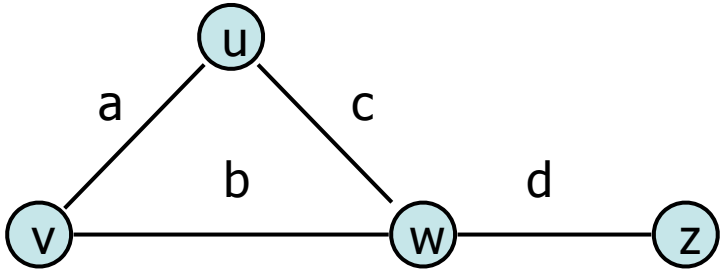
removeVertex(Vertex v):





Graph Implementation: Adjacency List

|V|= n, |E|= m



What's wrong with our implementation?

How can we fix it?