Data Structures

Graph Implementations

CS 225

March 30, 2026

Brad Solomon

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Exam 4 on 4/6 — 4/8

Autograded MC and one coding question

Manually graded short answer prompt

Practice exam releases this week on PL

Topics covered can be found on website

Register now!

https://courses.engr.illinois.edu/cs225/exams/

Lab this week is exam 4 review session

Like exam 2, some new questions will be released for lab

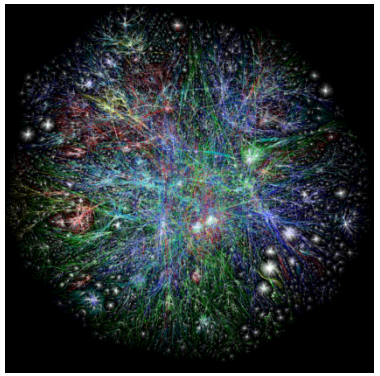
A great opportunity to go over practice exam too!

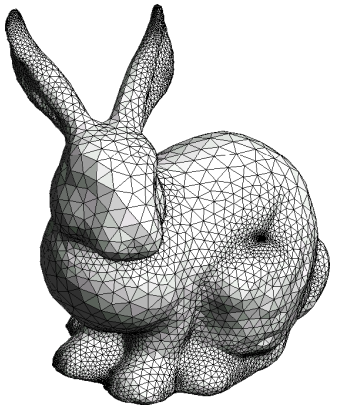
Form a study group to peer grade FRQs

Learning Objectives

Discuss graph implementation and storage strategies

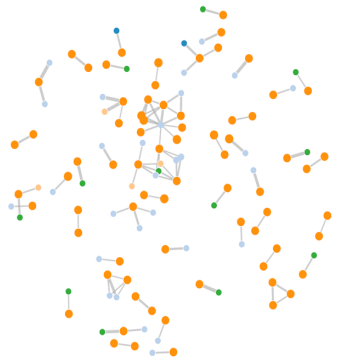
Graphs





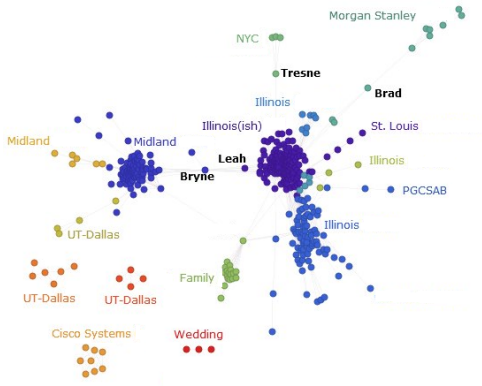
To study all of these structures:

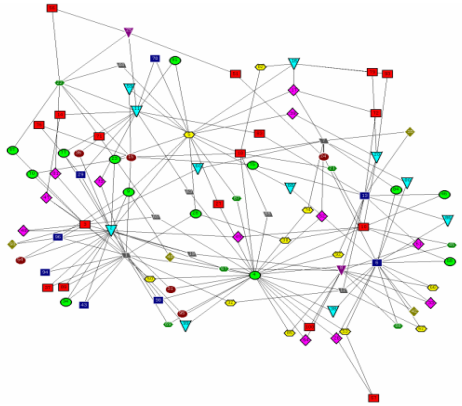
1. A common vocabulary
2. Graph implementations
3. Graph traversals
4. Graph algorithms

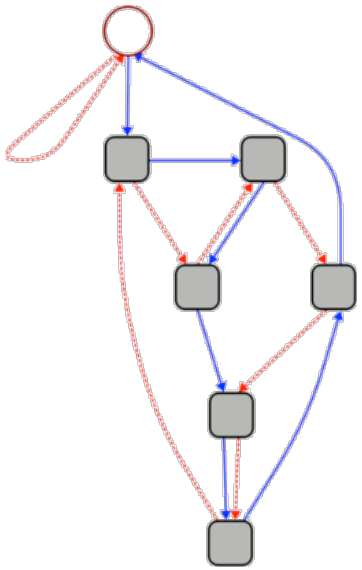


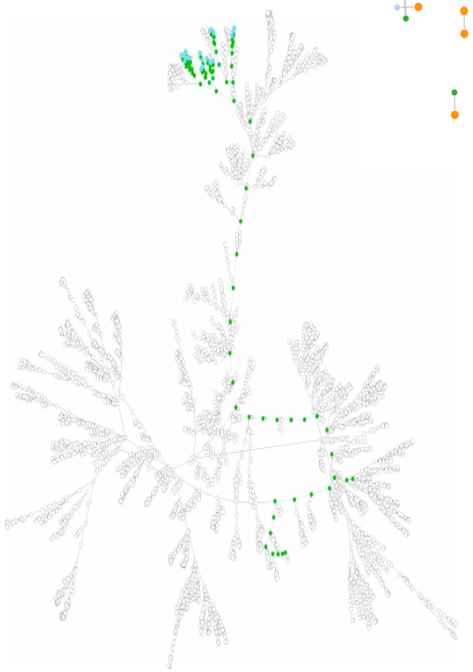
HAMLET

TROILUS AND CRESSIDA





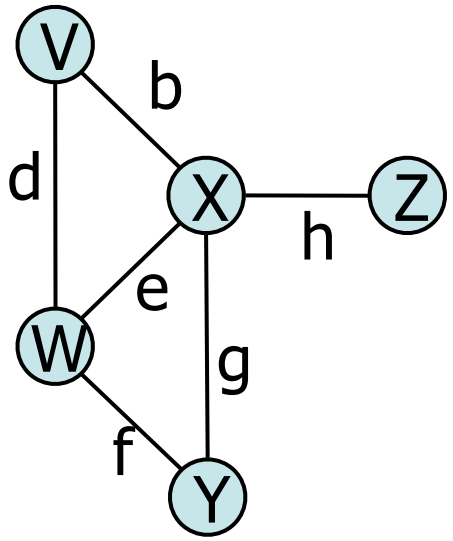




Graph ADT

Data:

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



Functions:

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

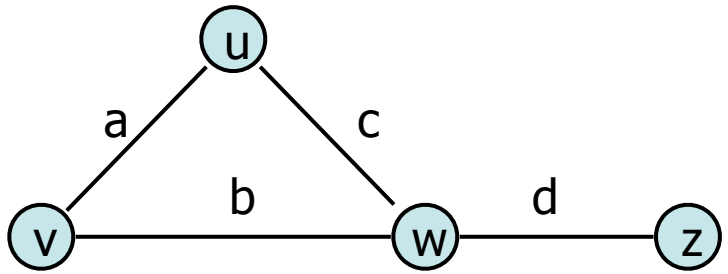
- removeVertex(Vertex v);
- removeEdge(Vertex v1, Vertex v2);

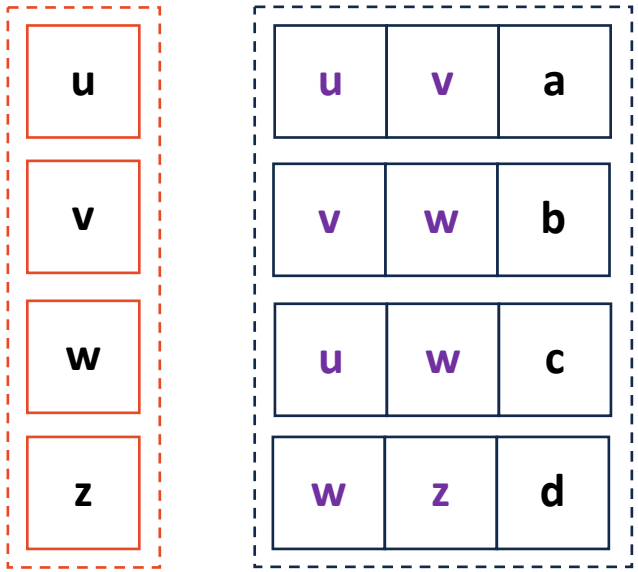
- incidentEdges(Vertex v);
- areAdjacent(Vertex v1, Vertex v2);

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

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

The equivalent of an 'unordered' data structure





Vertex Storage:

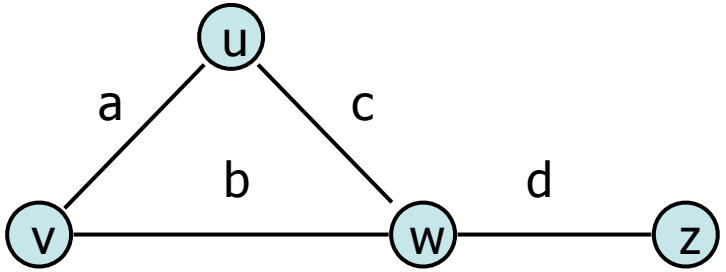
An optional list of vertices

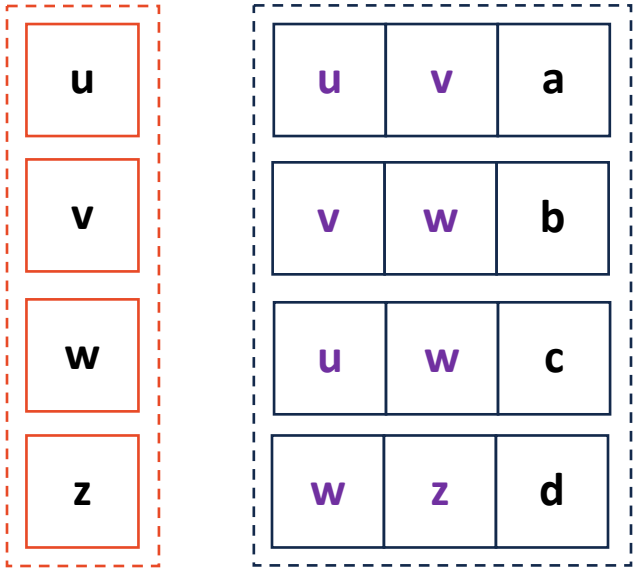
Edge Storage:

A list storing edges as (V1, V2, Weight)

Most graphs are stored as just an edge list!

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





getEdges(Vertex v)

Search edge storage for ‘v’

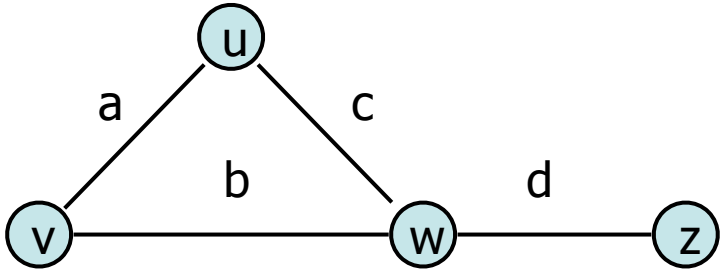
Since unsorted array: O(m)

areAdjacent(Vertex v1, Vertex v2)

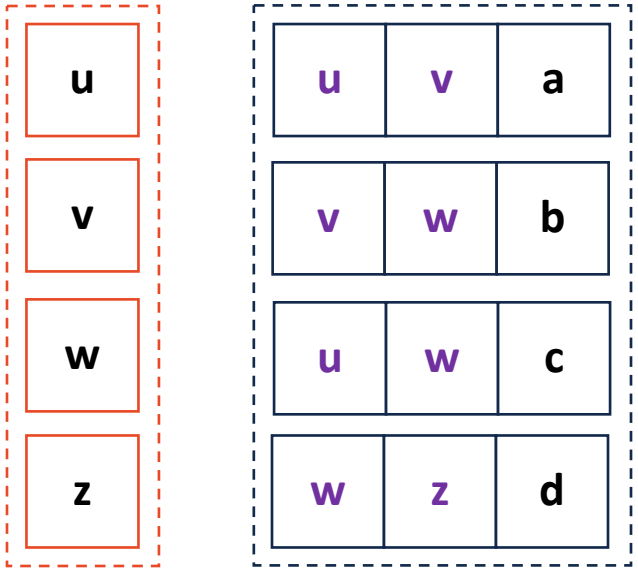
Search edge storage for ‘v1’ AND ‘v2’

Since unsorted array: O(m)

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

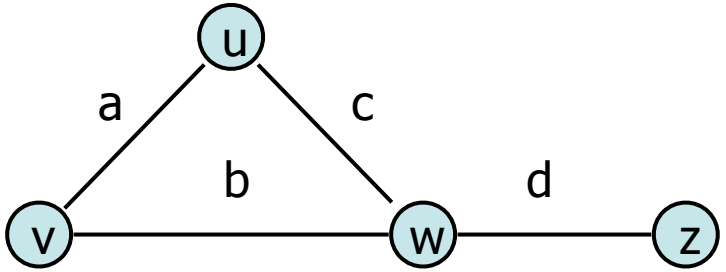


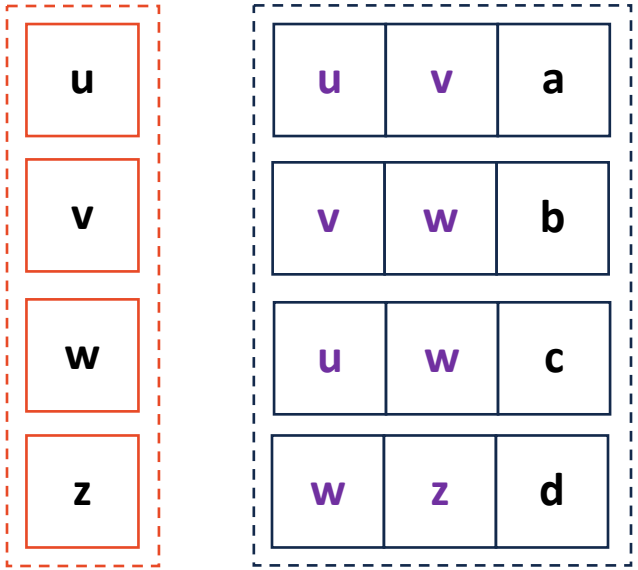
insertVertex(K key)



removeVertex(Vertex v)

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





insertVertex(K key)

Add ‘key’ to vertex array

Since unsorted array: O(1)*

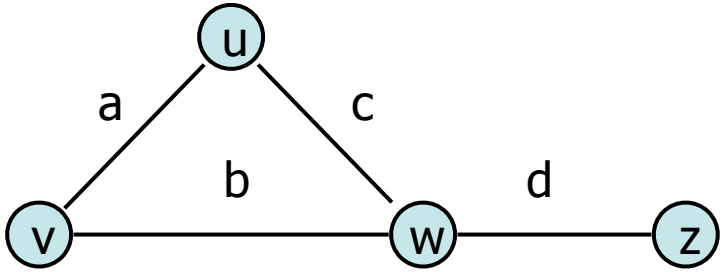
removeVertex(Vertex v)

Remove vertex from vertex array

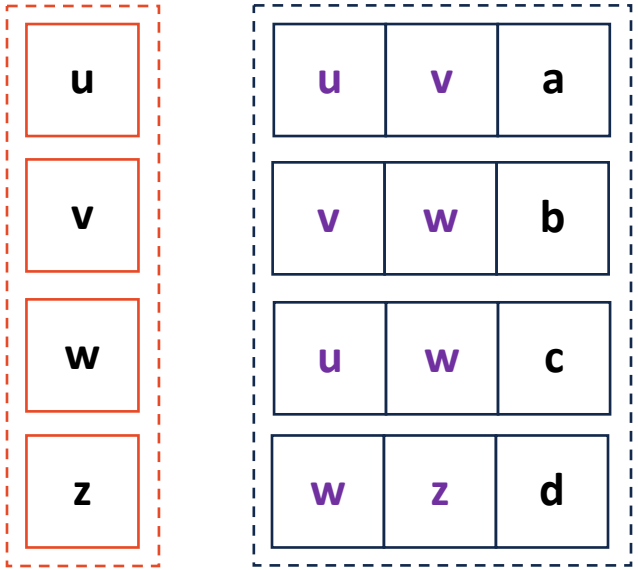
Remove vertex from edge array

Since unsorted array: O(n+m)

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

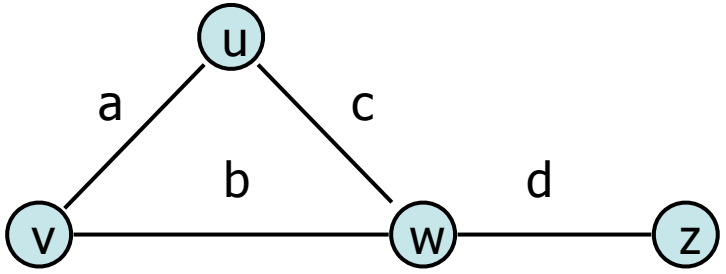


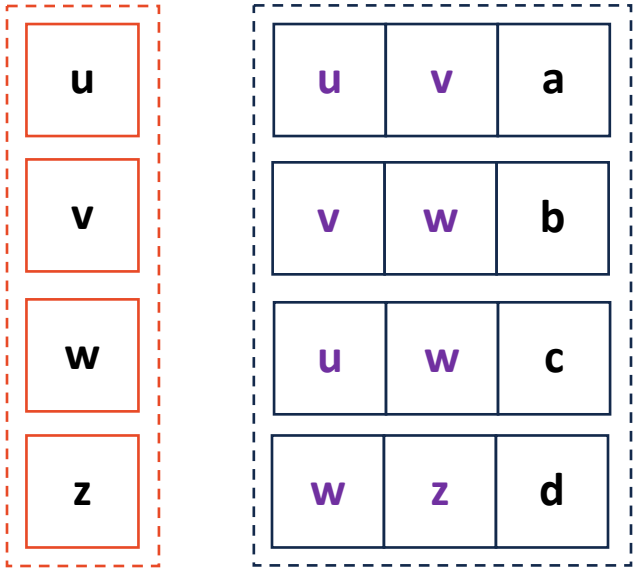
insertEdge(Vertex v1, Vertex v2, K key)



removeEdge(Vertex v1, Vertex v2)

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





insertEdge(Vertex v1, Vertex v2, K key)

Tricky! Can we assume edge doesn't exist?

If yes — easy: O(1)*

If no — still easy but slower: O(m)

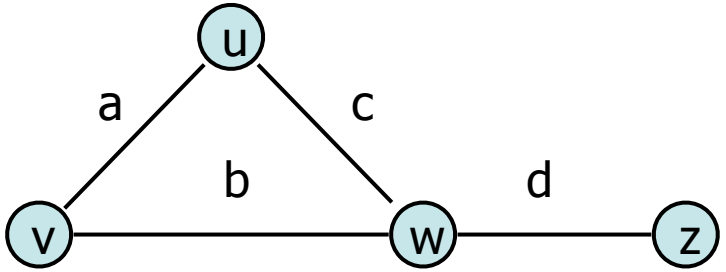
removeEdge(Vertex v1, Vertex v2)

Search edge array for 'v1' and 'v2'

Remove it if it exists

Since unsorted array: O(m)

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



insertVertex(K key)

insertEdge(Vertex v1, Vertex v2, K key)

u
v
w
z

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

incidentEdges(Vertex v)

areAdjacent(Vertex v1, Vertex v2)

removeVertex(Vertex v)

removeEdge(Vertex v1, Vertex v2)

Graph Implementation: Edge List

Pros:

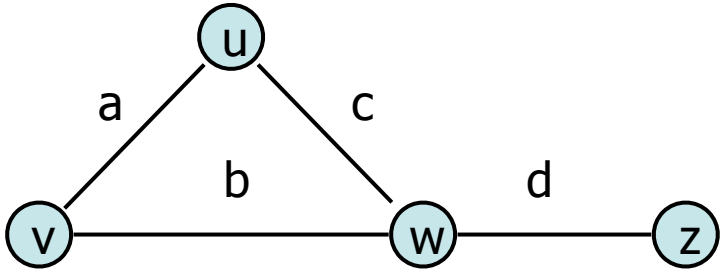
Cons:

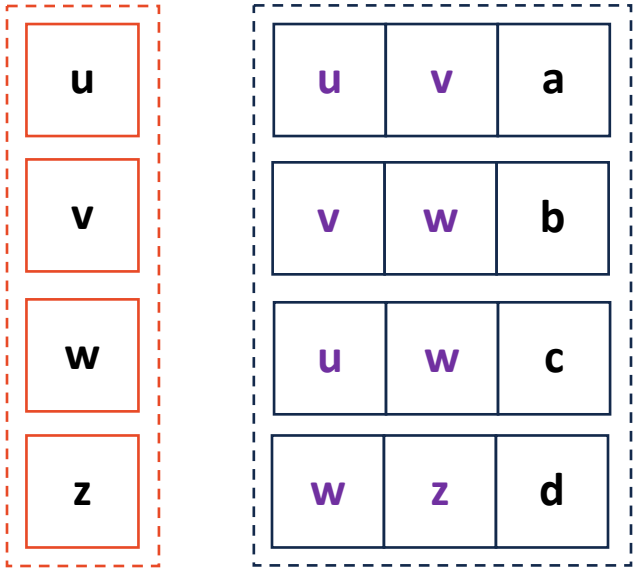
Graph Implementation: Brainstorming better

What operations might I want to do very quickly?

What modifications might allow me to do these things faster?

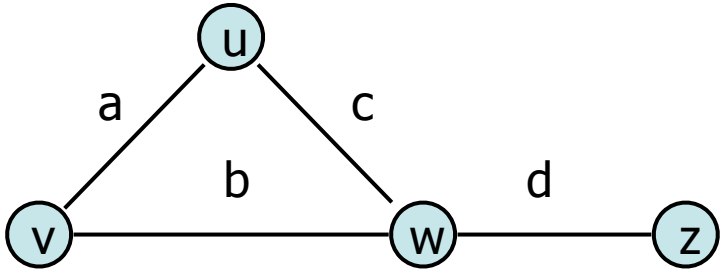
Graph Implementation: Adjacency Matrix





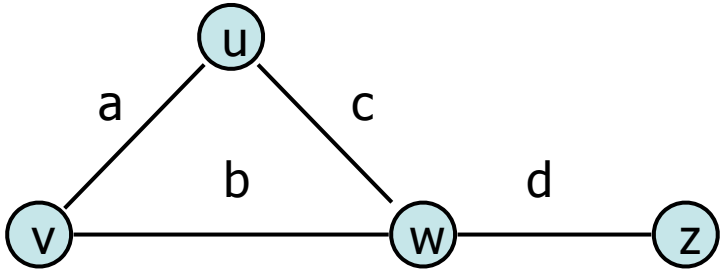
	u	v	w	z
u				
v				
w				
z				

Graph Implementation: Adjacency Matrix



	u	v	w	z
u				
v				
w				
z				

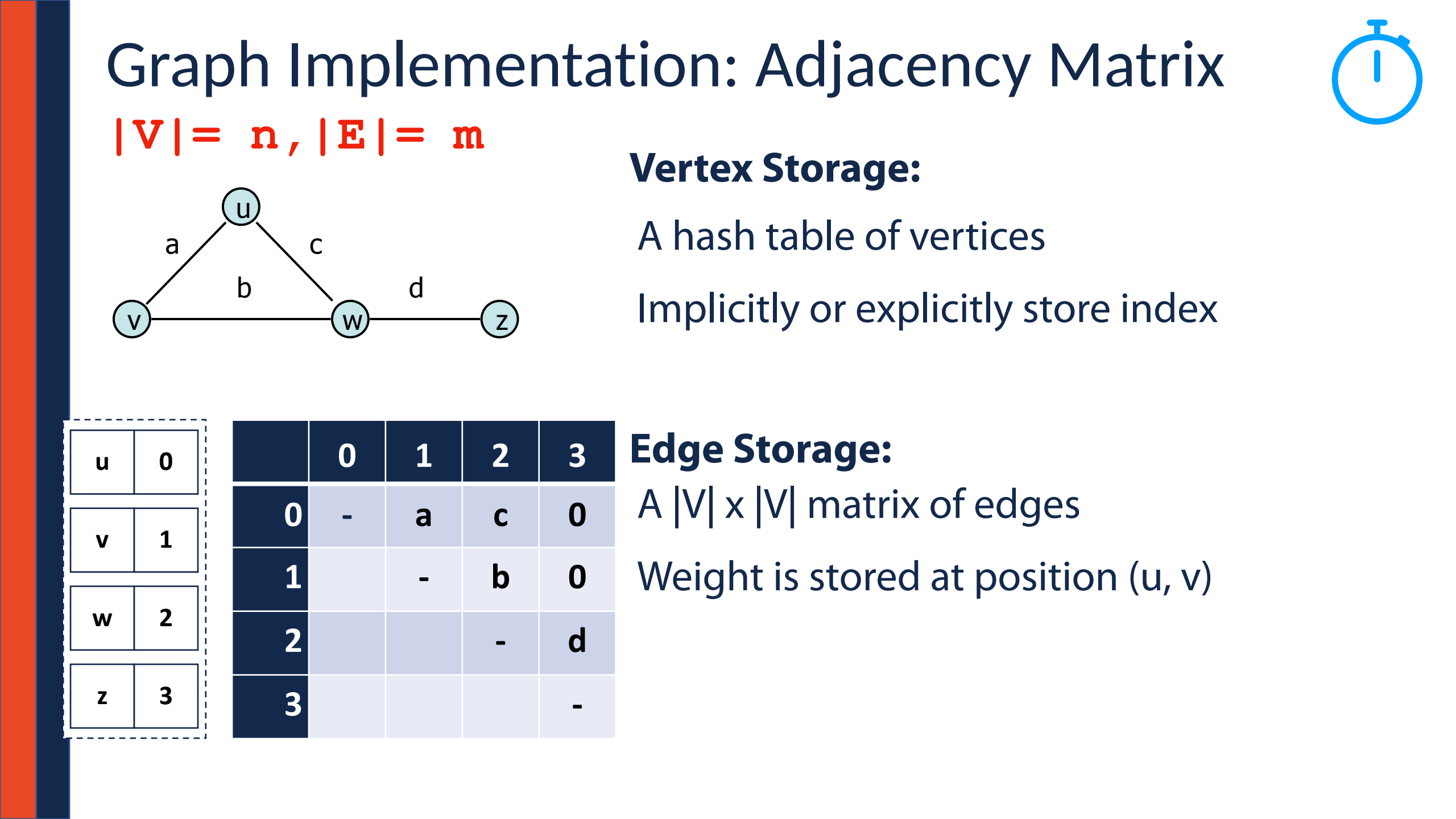
Graph Implementation: Adjacency Matrix



u	0
v	1
w	2
z	3

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

Graph Implementation: Adjacency Matrix



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

Vertex Storage:

A hash table of vertices

Implicitly or explicitly store index

u	0
v	1
w	2
z	3

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

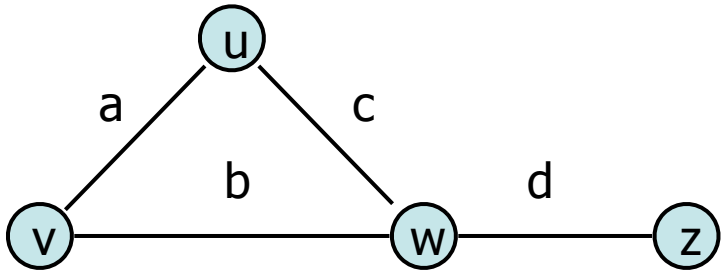
Edge Storage:

A |V| x |V| matrix of edges

Weight is stored at position (u, v)

Graph Implementation: Adjacency Matrix

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



incidentEdges(Vertex v):

u	0
v	1
w	2
z	3

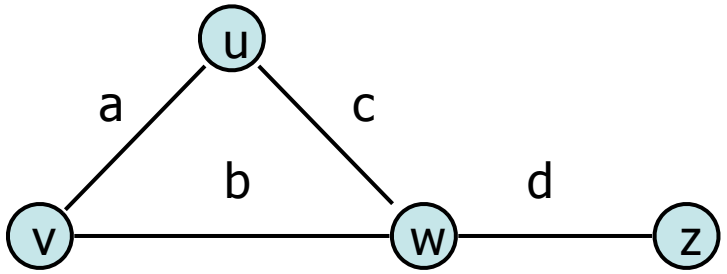
	0	1	2	3
0	-	a	c	0
1		-	b	0
2			-	d
3				-

areAdjacent(Vertex v1, Vertex v2):

Graph Implementation: Adjacency Matrix

Join Code: 225

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



u	0
v	1
w	2
z	3

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

incidentEdges(Vertex v):

Look up row (or column) in matrix

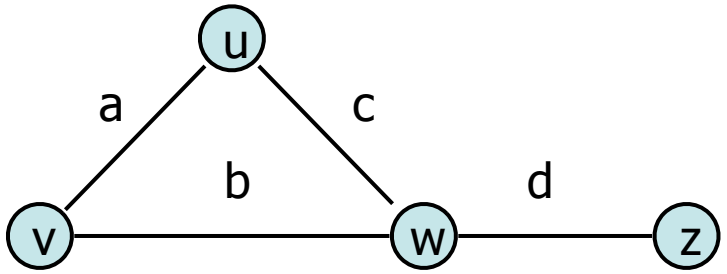
areAdjacent(Vertex v1, Vertex v2):

Look up row (and column) in matrix

What are my Big Os?

Graph Implementation: Adjacency Matrix

|V|= n, |E|= m



u	0
v	1
w	2
z	3

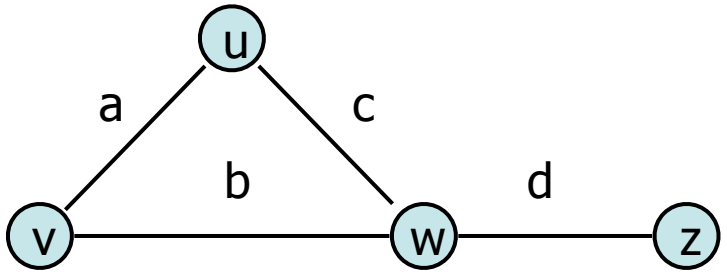
	0	1	2	3
0	-	a	c	0
1		-	b	0
2			-	d
3				-

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

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

Graph Implementation: Adjacency Matrix

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



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):

Look up row and column

Change value

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

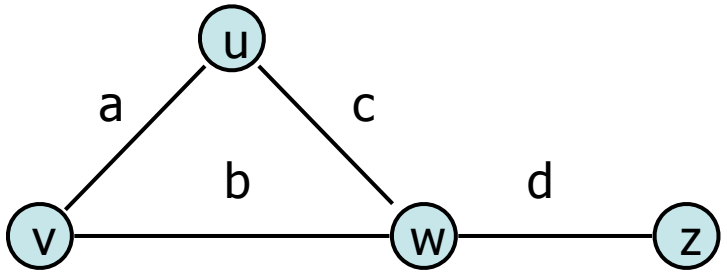
Look up row and column

Change value

What are my Big Os?

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				-

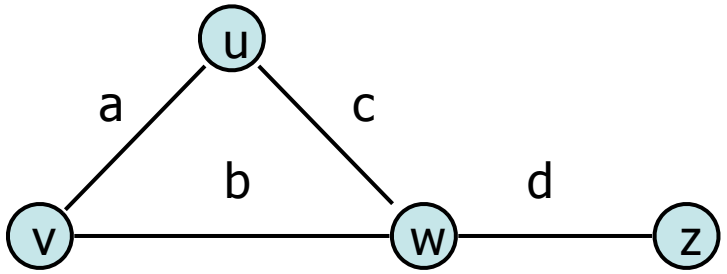
insertVertex(K key):

removeVertex(Vertex v):

Graph Implementation: Adjacency Matrix

Join Code: 225

|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):

Insert new key to vertex mapping

Add a new row and column to matrix

removeVertex(Vertex v):

Remove v from vertex mapping

Remove an entire row / entire column

What are my Big Os?

Graph Implementation: Adjacency Matrix

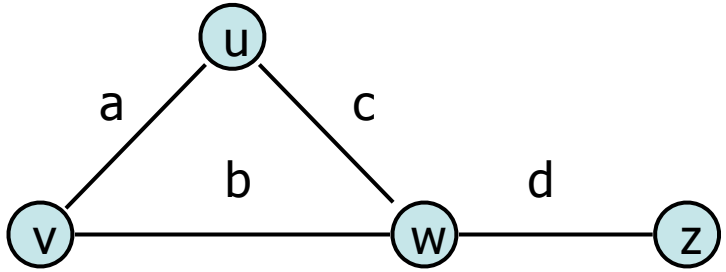
Pros:

Cons:

Graph Implementation Brainstorming

We want something…

Faster than an edge list

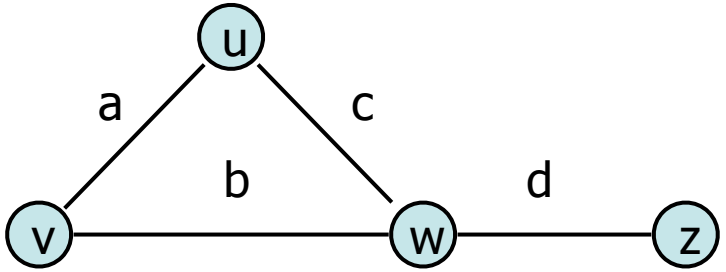


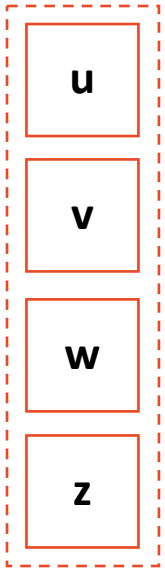
Less space than an adjacency matrix

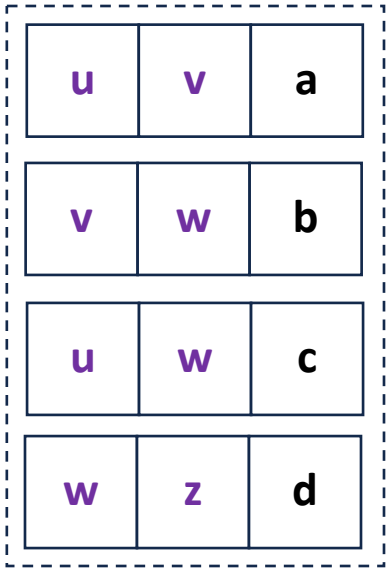
Particularly good at **finding all adjacent elements (neighbors)**

Graph Implementation: Edge List + ?

|V|= n,|E|= m

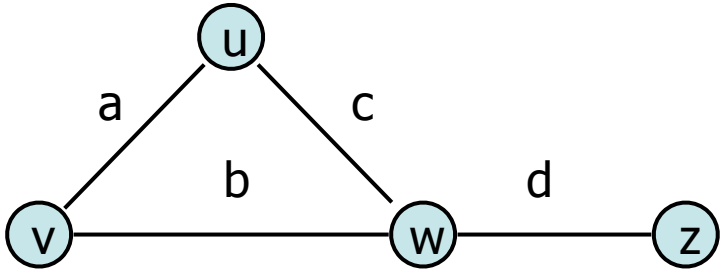


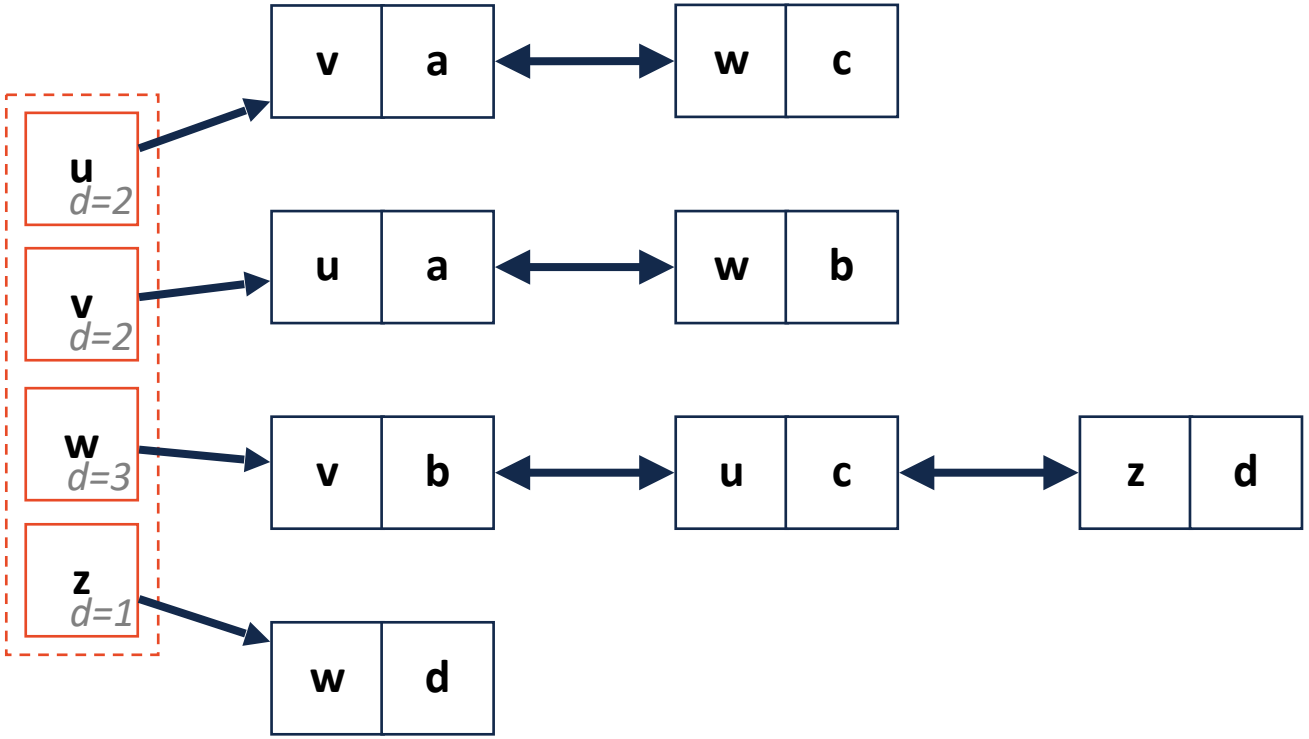




Graph Implementation: Adjacency List

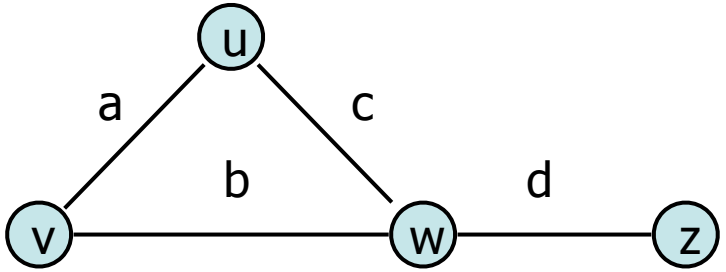
|V|= n, |E|= m



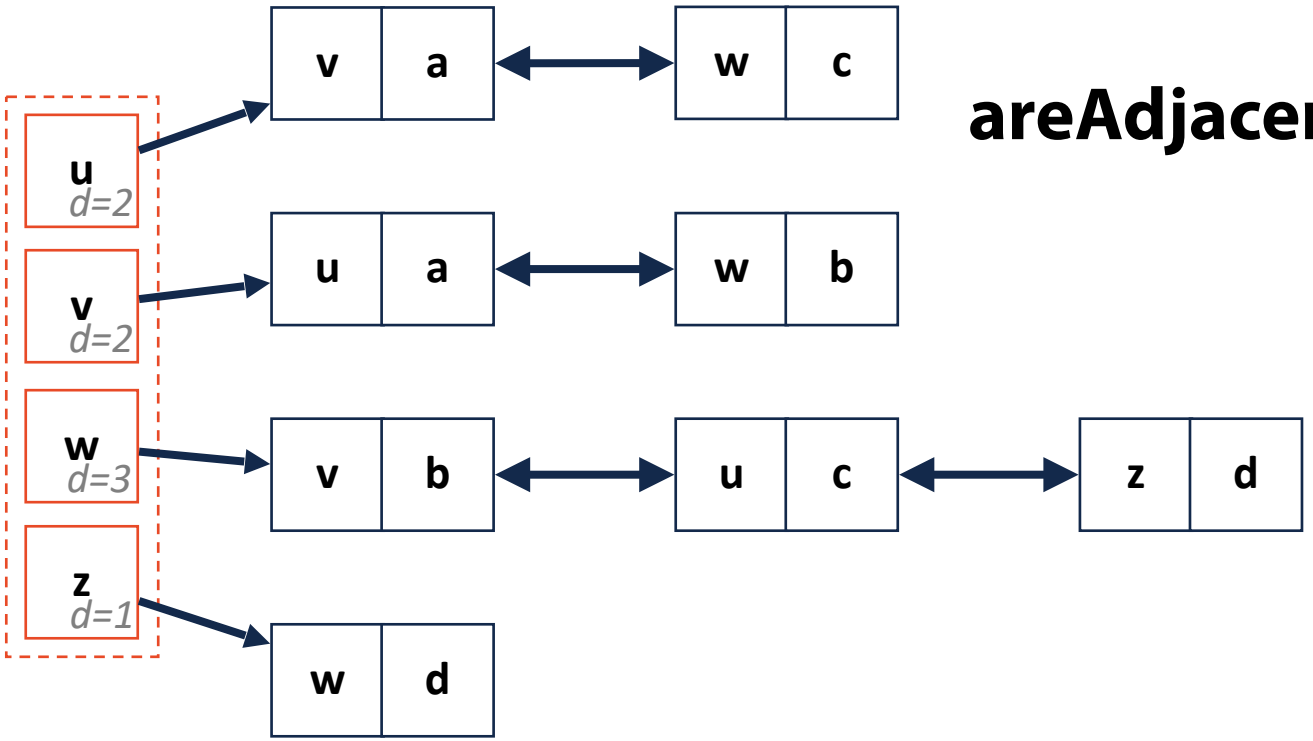


Graph Implementation: Adjacency List

|V|= n, |E|= m



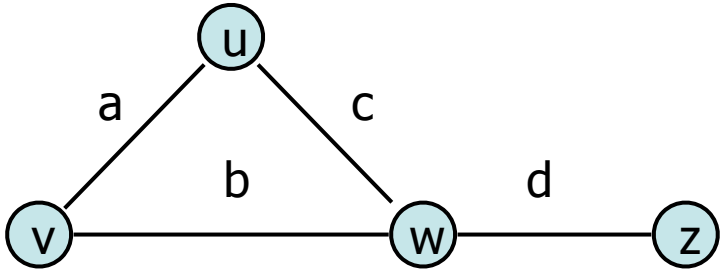
incidentEdges(Vertex v):



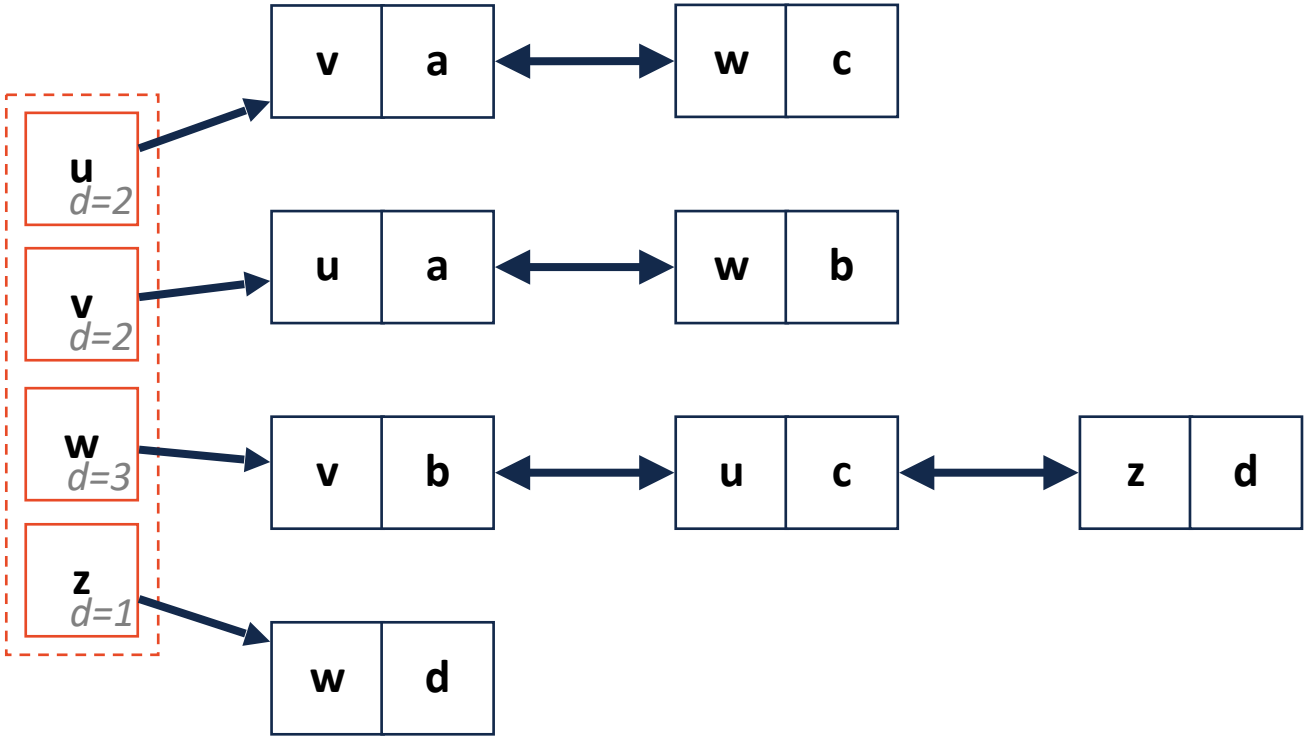
areAdjacent(Vertex v1, Vertex v2):

Graph Implementation: Adjacency List

|V|= n, |E|= m



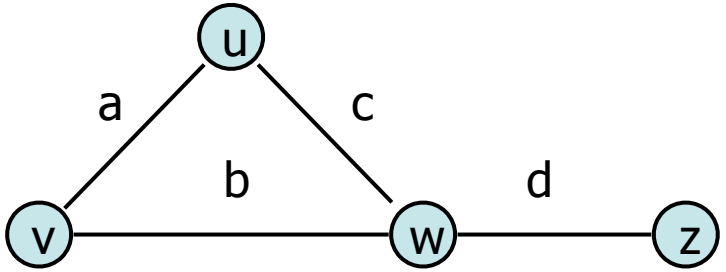
removeEdge(Vertex v1, Vertex v2, K key):

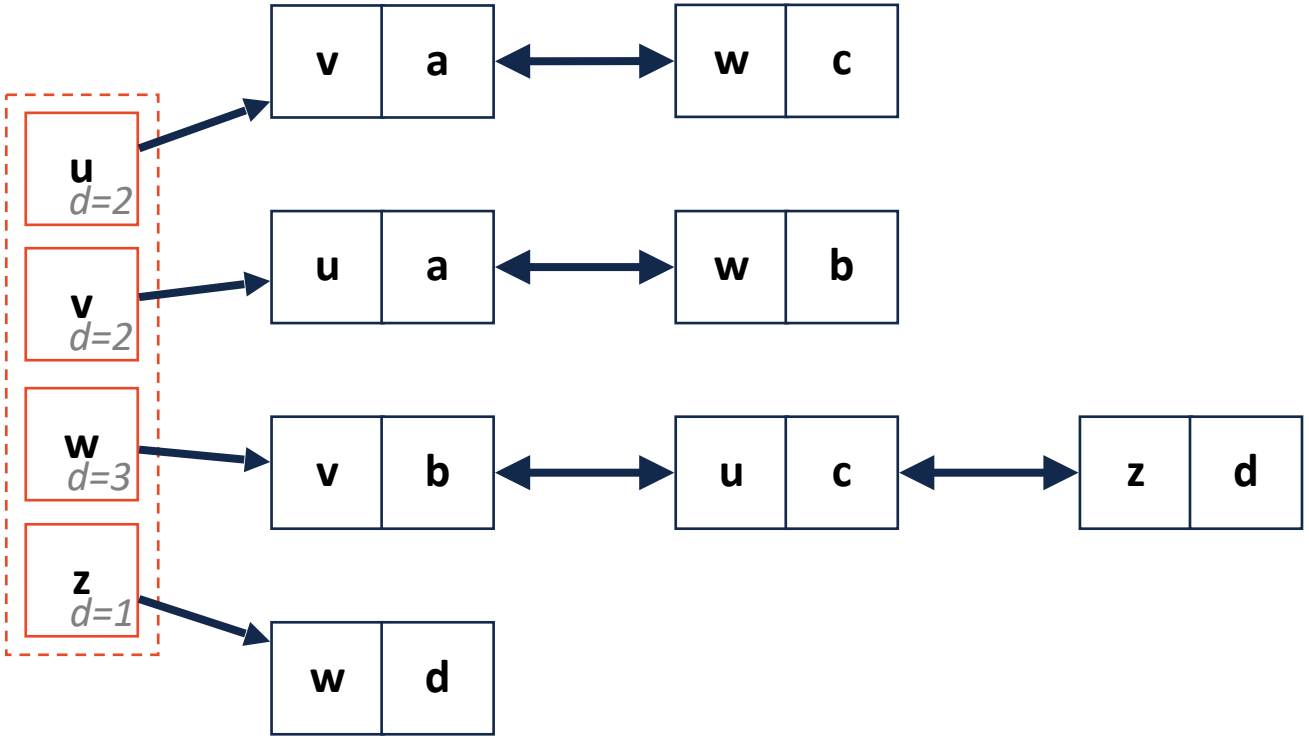


Graph Implementation: Adjacency List

|V|= n, |E|= m

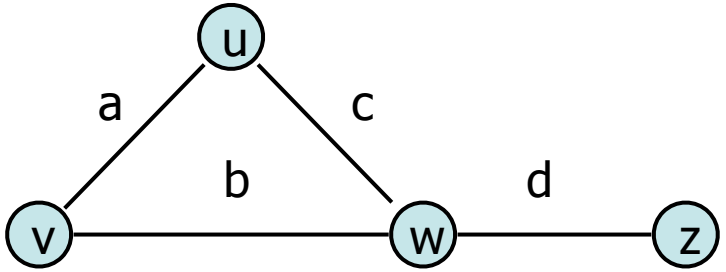
removeVertex(Vertex v):





Graph Implementation: Adjacency List

|V|= n, |E|= m



What's wrong with our implementation?

How can we fix it?