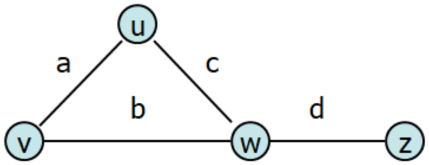
#34: Graph Implementation

April 13 2018 · *Wade Fagen-Ulmschneider*

Graph Implementation #1: Edge List

- HashTable storage of our vertex set
- List storage of our edge set
- O(1) runtime: insertVertex
- O(m) runtime: removeVertex, areAdjacent, and incidentEdges

Graph Implementation #2: Adjacency Matrix



Vert.
u
v
w
z

Edges		
		a
		b
		c
		d

Adj. Matrix				
▨	u	v	w	z
u				
v				
w				
z				

Operations on an Adjacency Matrix:

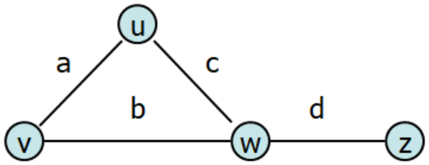
insertVertex(K key):

removeVertex(Vertex v):

areAdjacent(Vertex v1, Vertex v2):

incidentEdges(Vertex v):

Graph Implementation #3: Adjacency List



Vertex List
u
v
w
z

Edges		
		a
		b
		c
		d

Operations on an Adjacency List:

insertVertex(K key):

removeVertex(Vertex v):

areAdjacent(Vertex v1, Vertex v2):

incidentEdges(Vertex v):

Running Times of Classical Graph Implementations

	Edge List	Adj. Matrix	Adj. List
Space	**n+m**	**n+m**	**n^2**
insertVertex	**1**	**n**	**1**
removeVertex	**m**	**n**	**deg(v)**
insertEdge	**1**	**1**	**1**
removeEdge	**1**	**1**	**1**
incidentEdges	**m**	**n**	**deg(v)**
areAdjacent	**m**	**1**	**min(deg(v), deg(w))**

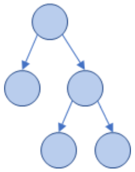
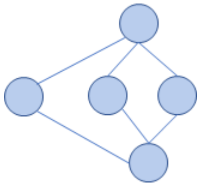
How do the algorithms compare?

...is one always better?

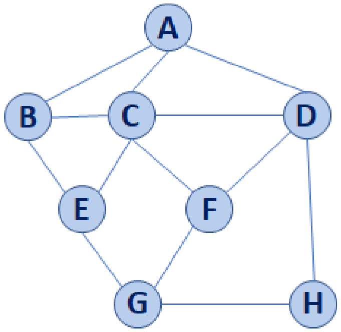
Graph Traversal

Objective: Visit every vertex and every edge in the graph.
Purpose: Search for interesting sub-structures in the graph.

We've seen traversal before – this is only slightly different:

BST	Graph
	

BST Graph Traversal



CS 225 – Things To Be Doing:

1. Topic list for Programming Exam C available; starts Tuesday 4/17
2. lab_puzzles ongoing; due Sunday, April 15th
3. MP6 due on Monday, April 16th
4. Daily POTDs are ongoing!