CS 225

#33: Graph Vocabulary + Implementation

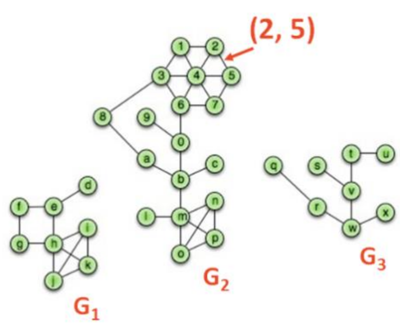
April 11, 2018 · *Wade Fagen-Ulmschneider*

Motivation:
Graphs are awesome data structures that allow us to represent an enormous range of problems. To study these problems, we need:

1. A common vocabulary to talk about graphs
2. Implementation(s) of a graph
3. Traversals on graphs
4. Algorithms on graphs

Graph Vocabulary

Consider a graph **G** with vertices **V** and edges **E**, **G=(V,E)**.



Incident Edges:
I(v) = { (x, v) in E }

Degree(v): **|I|**

Adjacent Vertices:
A(v) = { x : (x, v) in E }

Path(G_2): Sequence of vertices connected by edges

Cycle(G_1): Path with a common begin and end vertex.

Simple Graph(G): A graph with no self loops or multi-edges.

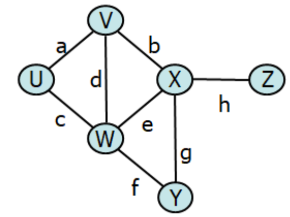
Subgraph(G): **G' = (V', E')**:
V' ∈ V, E' ∈ E, and (u, v) ∈ E → u ∈ V', v ∈ V'

Graphs that we will study this semester include:
Complete subgraph(G)
Connected subgraph(G)
Connected component(G)
Acyclic subgraph(G)
Spanning tree(G)

Size and Running Times

Running times are often reported by **n**, the number of vertices, but often depend on **m**, the number of edges.

For arbitrary graphs, the minimum number of edges given a graph that is:



Not Connected:

Minimally Connected:*

The maximum number of edges given a graph that is:

Simple:

Not Simple:

The relationship between the degree of the graph and the edges:

Proving the Size of a Minimally Connected Graph

Theorem: Every minimally connected graph **G=(V, E)** has **|V|-1** edges.

<u>Proof of Theorem</u>
Consider an arbitrary, minimally connected graph **G=(V, E)**.

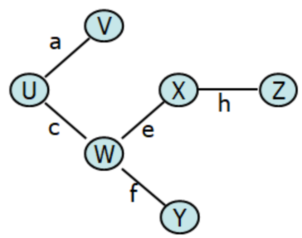
Lemma 1: Every connected subgraph of **G** is minimally connected.
(Easy proof by contradiction left for you.)

Inductive Hypothesis: For any **j** < |**V**|, any minimally connected graph of **j** vertices has **j-1** edges.

Suppose |V| = 1:
Definition: A minimally connected graph of 1 vertex has 0 edges.
Theorem: |V|-1 edges ➔ 1-1 = 0.

Suppose |V| > 1:
Choose any vertex **u** and let **d** denote the degree of **u**.

Remove the incident edges of **u**, partitioning the graph into ____ components: $\mathbf{C_0 = (V_0, E_0)}, \ldots, \mathbf{C_d = (V_d, E_d)}$.



By Lemma 1, every component $\mathbf{C_k}$ is a minimally connected subgraph of **G**.

By our ________________________:

Finally, we count edges:

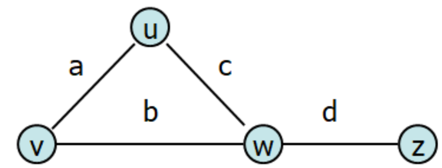
Graph ADT

Data	Functions

Vertices
 	
Edges
 	
Some data structure maintaining the structure between vertices and edges.
	`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 #1: Edge List

Vert.	Edges		
u			**a**
v			**b**
w			**c**
z			**d**



Operations:
insertVertex(K key):

removeVertex(Vertex v):

areAdjacent(Vertex v1, Vertex v2):

incidentEdges(Vertex v):

Graph Implementation #2: Adjacency Matrix

Vert.	Edges			Adj. Matrix	u	v	w	z
u			**a**	**u**				
v			**b**	**v**				
w			**c**	**w**				
z			**d**	**z**				

CS 225 – Things To Be Doing:

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