CS 225

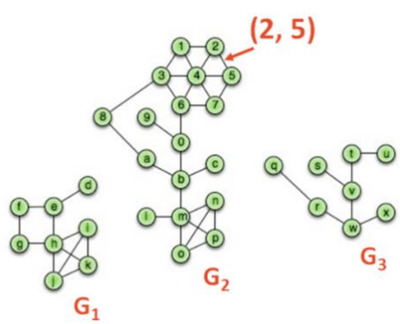
**#33: Graph Vocabulary + Implementation**
November 12, 2018 · *Wade Fagen-Ulmschneider*

**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'

---

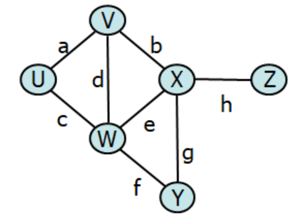
Graphs that we will study this semester include:
Complete subgraph(G)
Connected subgraph(G)
Connected component(G)
Acyclic subgraph(G)
Spanning tree(G)

**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 <u>**minimum**</u> number of edges given a graph that is:

*Not Connected:*

*Minimally Connected*:*




---

The <u>**maximum**</u> 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.)*

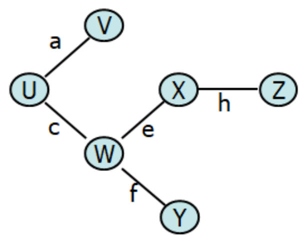
**<u>Suppose |V| = 1:</u>**

**Definition:**

**Theorem:**

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

**<u>Suppose |V| > 1:</u>**




1. Choose any vertex:
   -
   -

2. Partitions:
   -
   -
   - $C_0$ **:=**
   - $C_k$**, k=[1…d] :=**

3. Count the edges:

$|E_G|$ =

*…by application of our IH and Lemma #1, every component* $C_k$ *is a minimally connected subgraph of* ***G****…*

$|E_G|$ =

## Graph ADT

| Data | Functions |
|---|---|
| 1. Vertices<br><br>2. Edges<br><br>3. Some data structure maintaining the structure between vertices and edges. | `insertVertex(K key);`<br>`insertEdge(Vertex v1, Vertex v2, K key);`<br><br>`removeVertex(Vertex v);`<br>`removeEdge(Vertex v1, Vertex v2);`<br><br>`incidentEdges(Vertex v);`<br>`areAdjacent(Vertex v1, Vertex v2);`<br><br>`origin(Edge e);`<br>`destination(Edge e);` |

---

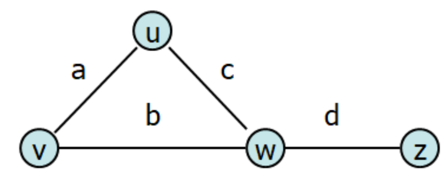
## Graph Implementation #1: Edge List

**Vert.**

| |
|---|
| **u** |
| **v** |
| **w** |
| **z** |

**Edges**

| | | |
|---|---|---|
| | | **a** |
| | | **b** |
| | | **c** |
| | | **d** |




**Operations:**

insertVertex(K key):

removeVertex(Vertex v):

areAdjacent(Vertex v1, Vertex v2):

incidentEdges(Vertex v):

| CS 225 – Things To Be Doing: |
|---|
| **1. Programming Exam C is different than usual schedule:**<br>Exam: Sunday, Dec 2 – Tuesday, Dec 4<br>**2.** lab_dict released this week; due on Tuesday, Nov. 27<br>**3.** MP6 EC+7 due tonight; final due date on Monday, Nov. 26<br>**4.** Daily POTDs are ongoing! |