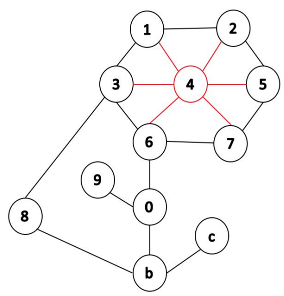
CS 225 Spring 2019 :: TA Lecture Notes
4/8 Graphs

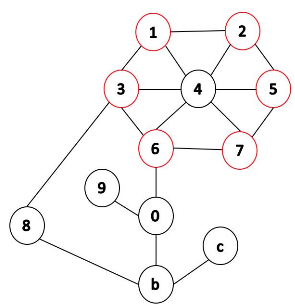
By Wenjie

Graph Vocabulary

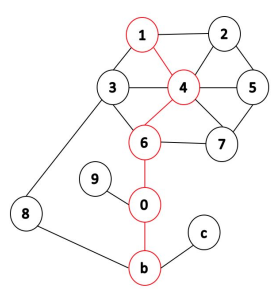
- size of the **vertices** |V| = n
- size of the **edges** |E| = m
- **Incident edges**: all edges that connected to that node.
  Example: incident edges to 4 are (4,1), (4,2), (4,3), (4,5), (4,6), and (4,7).



- **Degree**: the number of incident edges.
  Example: the degree of vertex 4 is 6 because it has 6 incident edges.
- **Adjacent vertex**: a vertex at the other end of the incident edge.
  Example: adjacent vertices to 4 are 1, 2, 3, 5, 6, and 7.



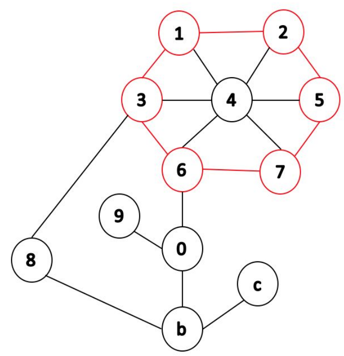
- **Path**: a sequence of vertices connected by edges.
  Example: a path from 1 to b includes visiting nodes 4, 6, and 0.



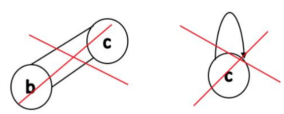
CS 225 Spring 2019 :: TA Lecture Notes
4/8 Graphs

By Wenjie

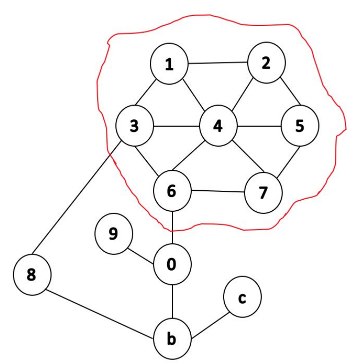
- **Cycle**: a path with common beginning and end.



- **Simple graph**: a graph with no self-loop edges and no multi-edges.



- **Subgraph**: any subset of vertices such that every edge in the subgraph implies that both vertices that are incident to that edge are part of that graph.
  Example: vertices {1, 2, 3, 4, 5, 6, 7} and edges {(4,1), (4,2), (4,3), (4,5), (4,6), (4,7), (1,2), (2,5), (5,7), (7,6), (6,3), (3,1)} construct a subgraph. Edges that are cut by the red line, (6,0) and (3,8), are not part of the subgraph.



Based on above terms we will see:

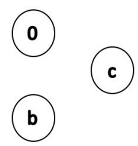
- **Complete subgraph**: every two distinct vertices are adjacent.
- **Connected subgraph**: there is a path between every two vertices in the graph.
- **Connected component**: a connected subgraph where non of the vertices are connected to the rest of the graph.

CS 225 Spring 2019 :: TA Lecture Notes
4/8 Graphs

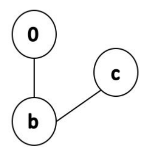
By Wenjie

- **Acyclic subgraphs**: a subgraph with no cycles.
- **Spanning trees**: a connected acyclic subgraph with minimal edge weight.

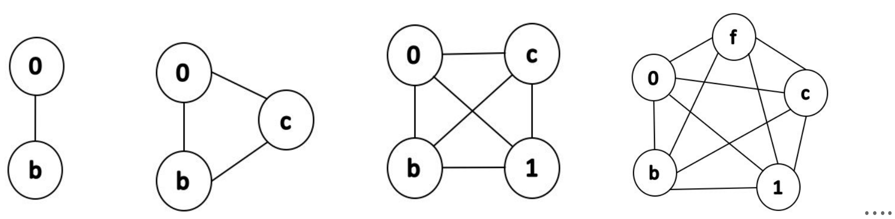
- **Minimal number of edges**:
  - Non-connected graph → m = 0



  - Connected graph → m = n - 1



- **Maximal number of edges**:
  - If the graph is not simple, number of edges: infinite.
  - If the graph is simple:



n	m
1	0
2	1
3	3
4	6
5	10
...	...

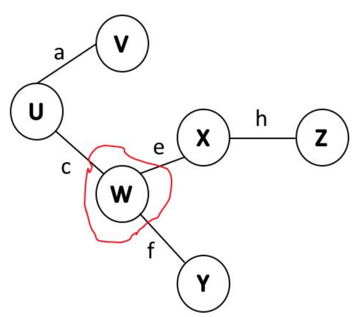
CS 225 Spring 2019 :: TA Lecture Notes
4/8 Graphs

By Wenjie

n	$\sum_{k=1}^{n-1} k = \quad O(n^2)$

- Sum of all degrees of all vertices → 2 * m

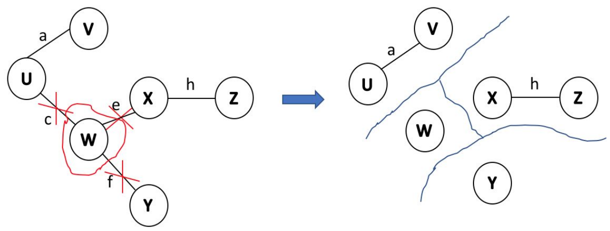
- **Theorem**: Every minimally connected graph has G=(V, E) has |V| - 1 edges.
    - **Lemma 1**: Every connected subgraph of G is minimally connected
        - We continue by assuming this lemma is true (proof is left for exercises)
- **Proof**
    - Consider an arbitrary minimally connected graph G=(E, V).
    - **Base case**: |V| = 1, by definition a minimally connected graph consisting of 1 vertex has 0 edges. By the theorem the number of edges should be |V| - 1 = (1 - 1) = 0.
    - **Inductive hypothesis**: For any j < |V|, any minimally connected graph of j vertices has (j - 1) edges.
    - **1. Suppose |V| > 1:**
        - Choose any vertex u and let d denote the degree of u.
    - Choose vertex w in the graph below.



        - **2. Partition**: remove the incident edges of u, partitioning the graph into (d + 1) components → $C_0 = (V_0, E_0)$, ..., $C_d = (V_d, E_d)$.
    - We choose vertex w, removed edges c, e, and f; and now we have 4 components → deg(w) + 1 = 3 + 1 = 4.

CS 225 Spring 2019 :: TA Lecture Notes
4/8 Graphs

By Wenjie



- By Lemma 1 every component C_k is a minimally connected subgraph of G.
- By our inductive hypothesis: $|E_k| = |V| - 1$.
- **3. Count edges**: $d + \sum_{k=1}^{d} E_k = d + \sum_{k=1}^{d} |V_k| = n - 1$. QED

Graph ADT

- **Data**: all vertices, all edges, and structure to maintain relations between vertices and edges.
- **Functions**:
  - insert vertex/edge
  - remove vertex/edge
  - find incident edges
  - check if two vertices are adjacent
  - find origin/destination.

Graph implementation 1 :: Edge List

- **Vertex collection**: Use hash table (find/remove/insert will be O(1)).
- **Edge collections**: Use a linked list (hash table is not good because we have many collisions (no random distribution, violates SUHA))
- **Running time**
  - **Insert vertex** → we are using hash table where insert takes O(1) time.
  - **Remove vertex** → removing from hash table takes O(1), but we need to remove incident edges which means we need to loop over edges list. We have m edges so it will take O(m)
  - **areAdjacent** → again, we need to loop over the edge list which takes O(m) time.
  - **InsertEdge** → add edge to edge list by adding to the front so it takes O(1)

CS 225 Spring 2019 :: TA Lecture Notes
4/8 Graphs

By Wenjie

- **incidentEdges** → O(m).
- The running times seem linear, however, we know that the relationship between number of nodes and the number of edges can be n^2; which means O(m) could in fact be O(n^2)