COL106: Data Structures and Algorithms

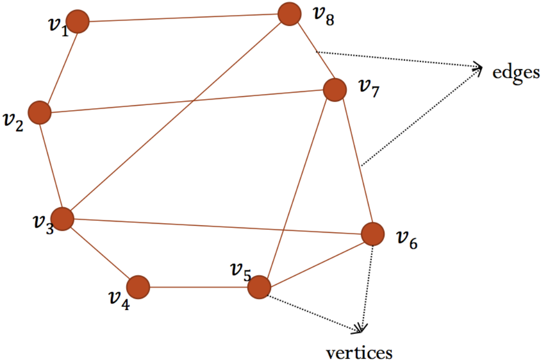
Ragesh Jaiswal, IIT Delhi

Course Overview

- Material that will be covered in the course:
  - Basic graph algorithms
  - Algorithm Design Techniques
    - Divide and Conquer
    - Greedy Algorithms
    - Dynamic Programming
    - Network Flows
  - Computational intractability

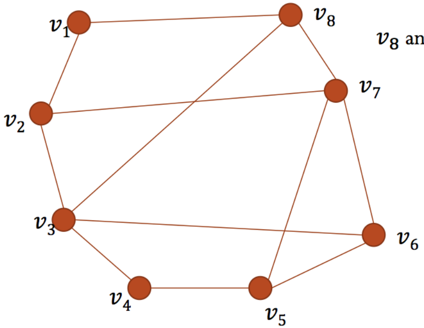
Graphs

- A way to represent a set of objects with pair-wise relationships among them.
- The objects are represented as vertices and the relationships are represented as edges.



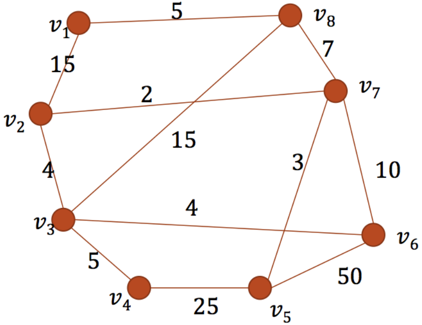
$$
\begin{aligned}
G &= (V, E) \\
V &= \{v_1, \ldots, v_8\} \\
E &= \{(v_1, v_8), \ldots\}
\end{aligned}
$$

- Examples
  - Social networks
  - Communication networks
  - Transportation networks
  - Dependency networks

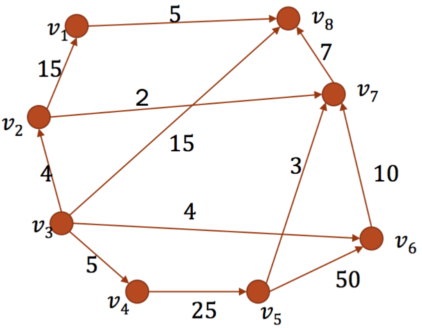


v_8 and v_7 are friends

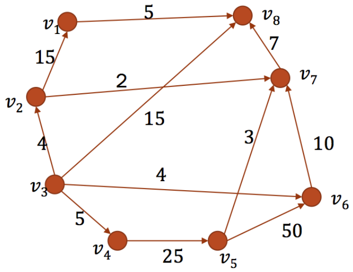
- Weighted graphs: There are weights associated with each edge quantifying the relationship. For example, delay in communication network.



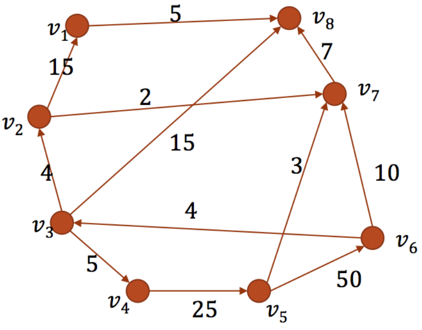
- Directed graphs: Asymmetric relationships between the objects. For example, one way streets.

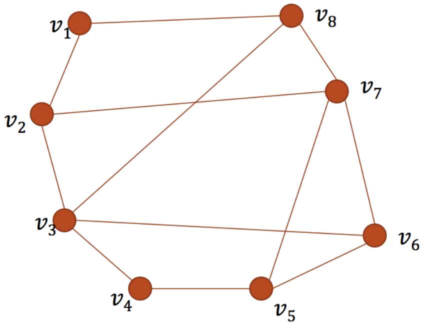


- Path: A sequence of vertices $v_1, v_2, ..., v_k$ such that for any consecutive pair of vertices v_i, v_{i+1}, (v_i, v_{i+1}) is an edge in the graph. It is called a path from v_1 to v_k.
  - Simple path: A simple path from vertex u to a different vertex v is a path that has distinct vertices.
- Cycle: A cycle is a path where $v_1 = v_k$ and $v_1, ..., v_{k-1}$ are distinct vertices.
- The above definitions are for directed graphs but generalise for undirected graphs.

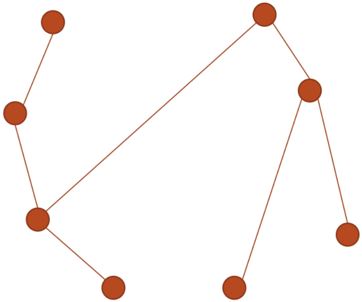


- Strongly connected: A graph is called strongly connected iff for any pair of vertices u, v, there is a path from u to v and a path from v to u.

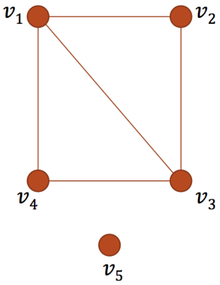




- Tree: A strongly connected, undirected graph is called a tree if it has no cycles.
- How many edges does a tree with n nodes have?

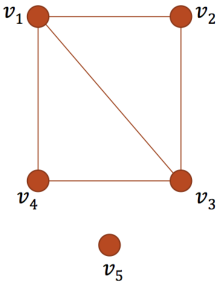


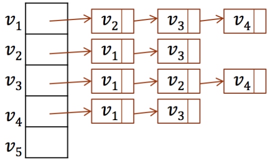
- Adjacency matrix: Store connectivity in a matrix.
- Space: $O(n^2)$



	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	1	0
v_2	1	0	1	0	0
v_3	1	1	0	1	0
v_4	1	0	1	0	0
v_5	0	0	0	0	0

- Adjacency list: For each vertex, store its neighbors.
- Space: $O(n+m)$





Graph Algorithms

Problem

Given an (undirected) graph $G = (V, E)$ and two vertices s, t, check if there is a path between s and t.

Problem

Given an (undirected) graph $G = (V, E)$ and two vertices s, t, check if there is a path between s and t.

- Alternate problem: What are the vertices that are reachable from s. Is t among these reachable vertices?
- This is also known as *graph exploration*. That is, explore all vertices reachable from a starting vertex s.
    - Breadth First Search (BFS)
    - Depth First Search (DFS)

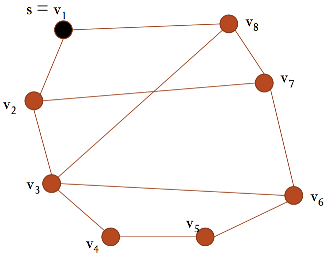
Breadth First Search (BFS)

BFS(G, s)

- $Layer(0) = \{s\}$
- $i \leftarrow 1$
- While(true)
  - Visit all new nodes that have an edge to a vertex in $Layer(i-1)$
  - Put these nodes in the set $Layer(i)$
  - If $Layer(i)$ is empty, then end
  - $i \leftarrow i+1$

Breadth First Search (BFS)

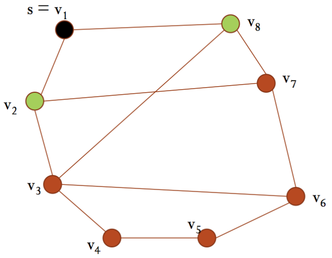
BFS(G, s)
- $Layer(0) = \{s\}$
- $i \leftarrow 1$
- While(true)
    - Visit all new nodes that have an edge to a vertex in $Layer(i-1)$
    - Put these nodes in the set $Layer(i)$
    - If $Layer(i)$ is empty, then end
    - $i \leftarrow i+1$



Breadth First Search (BFS)

BFS(G, s)

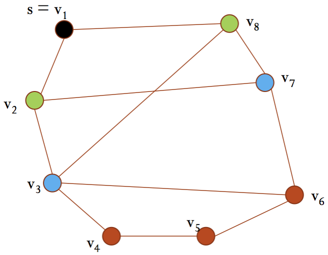
- $Layer(0) = \{s\}$
- $i \leftarrow 1$
- While(true)
    - Visit all new nodes that have an edge to a vertex in $Layer(i-1)$
    - Put these nodes in the set $Layer(i)$
    - If $Layer(i)$ is empty, then end
    - $i \leftarrow i + 1$



Breadth First Search (BFS)

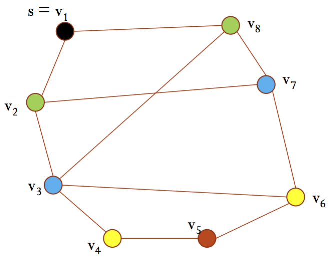
BFS(G, s)

- $Layer(0) = \{s\}$
- $i \leftarrow 1$
- While(true)
    - Visit all new nodes that have an edge to a vertex in $Layer(i-1)$
    - Put these nodes in the set $Layer(i)$
    - If $Layer(i)$ is empty, then end
    - $i \leftarrow i + 1$



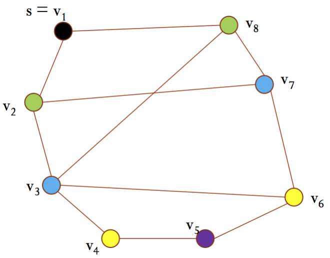
Breadth First Search (BFS)

BFS(G, s)
- $Layer(0) = \{s\}$
- $i \leftarrow 1$
- While(true)
  - Visit all new nodes that have an edge to a vertex in $Layer(i-1)$
  - Put these nodes in the set $Layer(i)$
  - If $Layer(i)$ is empty, then end
  - $i \leftarrow i+1$



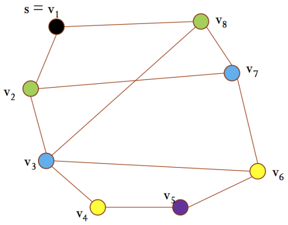
Breadth First Search (BFS)

BFS(G, s)
- $Layer(0) = \{s\}$
- $i \leftarrow 1$
- While(true)
  - Visit all new nodes that have an edge to a vertex in $Layer(i-1)$
  - Put these nodes in the set $Layer(i)$
  - If $Layer(i)$ is empty, then end
  - $i \leftarrow i + 1$



Breadth First Search (BFS)

BFS(G, s)
- $Layer(0) = \{s\}$
- $i \leftarrow 1$
- While(true)
  - Visit all new nodes that have an edge to a vertex in $Layer(i-1)$
  - Put these nodes in the set $Layer(i)$
  - If $Layer(i)$ is empty, then end
  - $i \leftarrow i+1$



- Theorem 1: The shortest path from s to any vertex in $Layer(i)$ is equal to i.

End