CSL 356: Analysis and Design of Algorithms

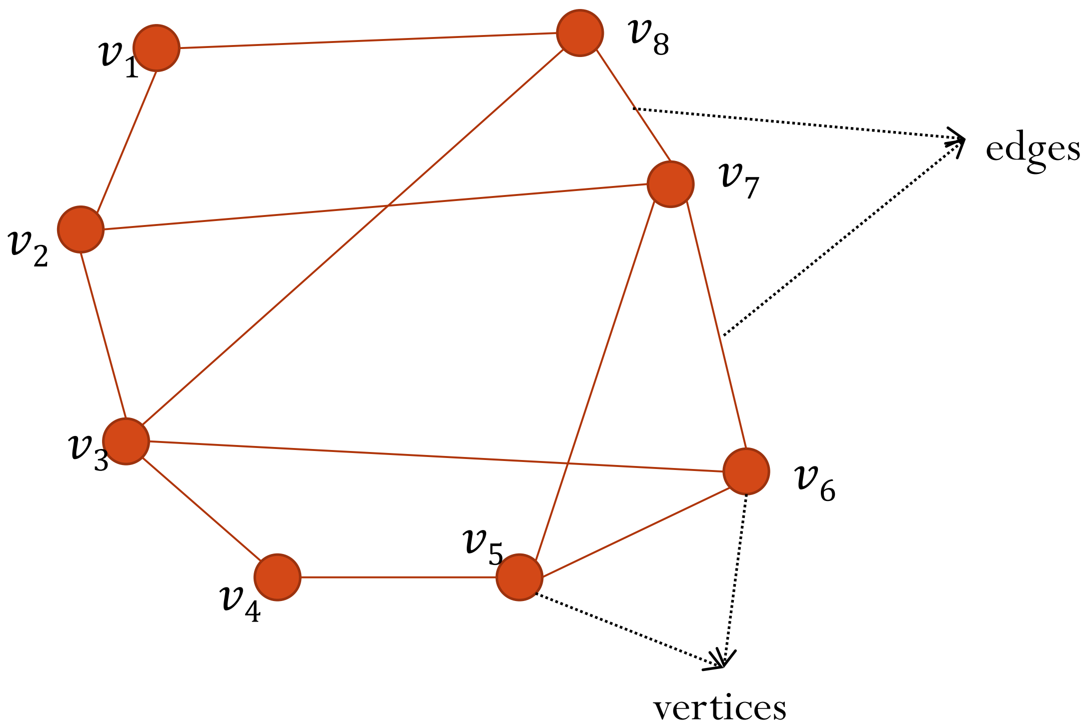
Ragesh Jaiswal

CSE, IIT Delhi

Graphs

Graphs: Introduction

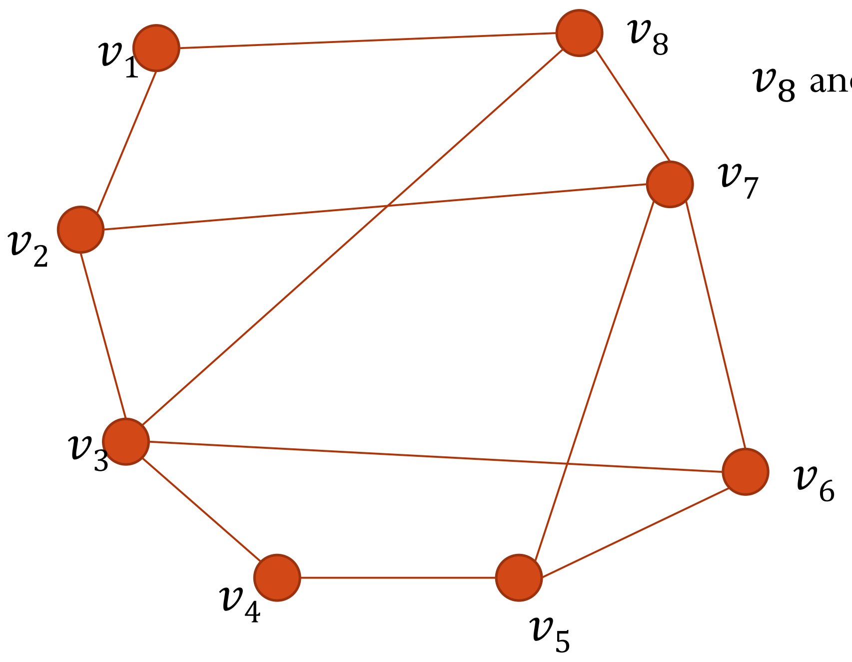
- A way to represent a set of objects with pair-wise relationships among them.
- The objects are represented as vertices and the relationships are represented as edges.



$$G = (V, E)$$
$$V = \{v_1, \dots, v_8\}$$
$$E = \{(v_1, v_8), \dots\}$$

Graphs: Introduction

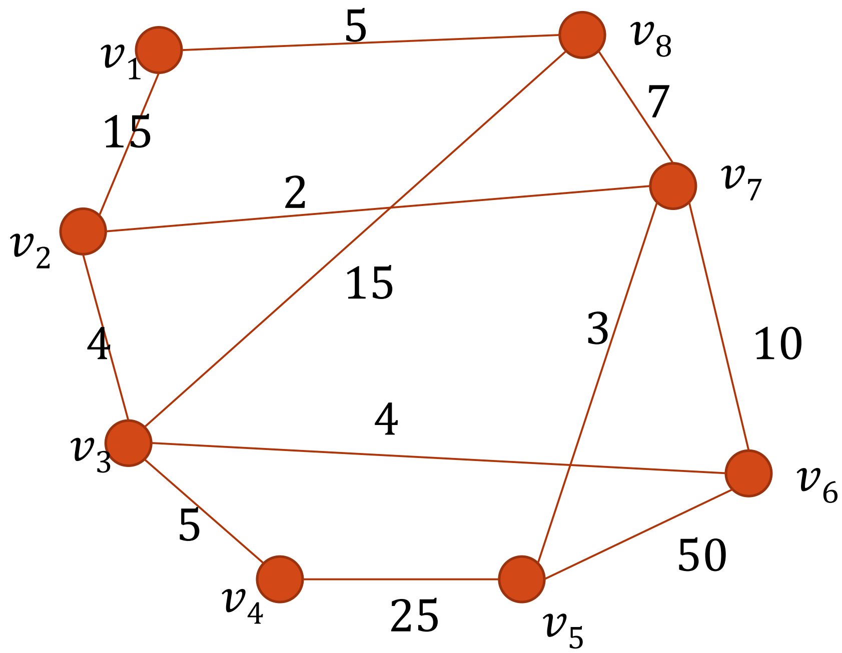
- Examples:
  - Social networks
  - Communication networks
  - Transportation networks
  - Dependency networks



v_8 and v_7 are friends

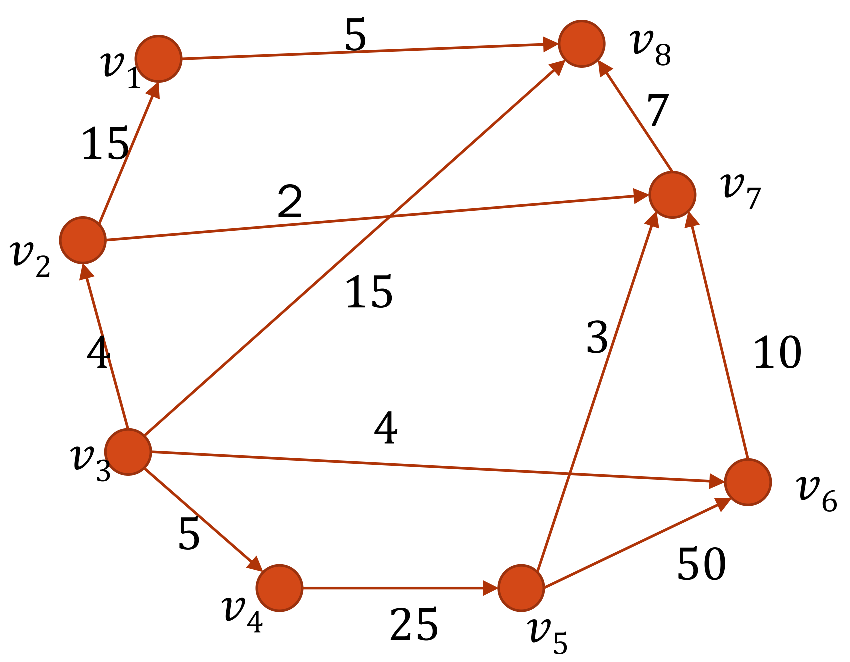
Graphs: Introduction

- <u>Weighted Graphs</u>: There are weights associated with each edge quantifying the relationship. For example, delay in communication network.



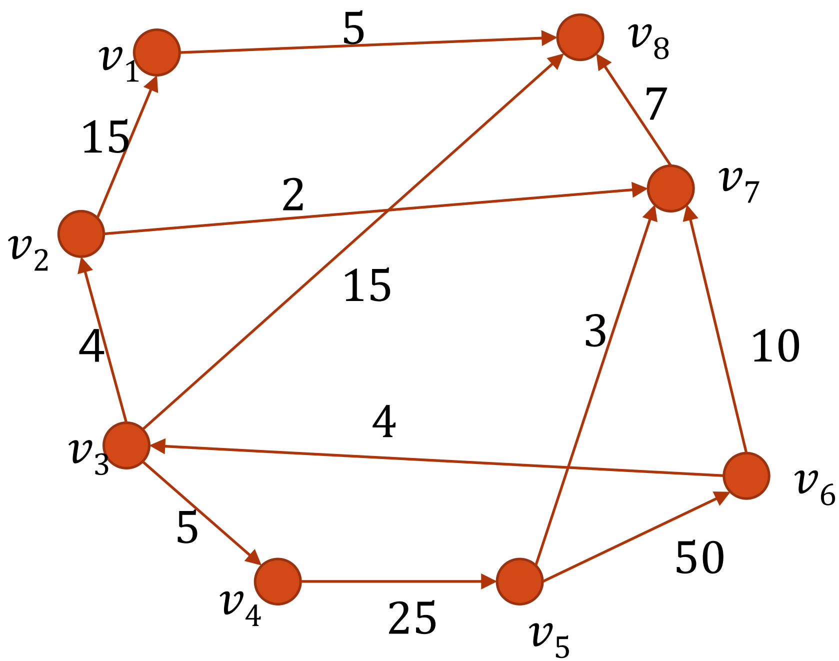
Graphs: Introduction

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



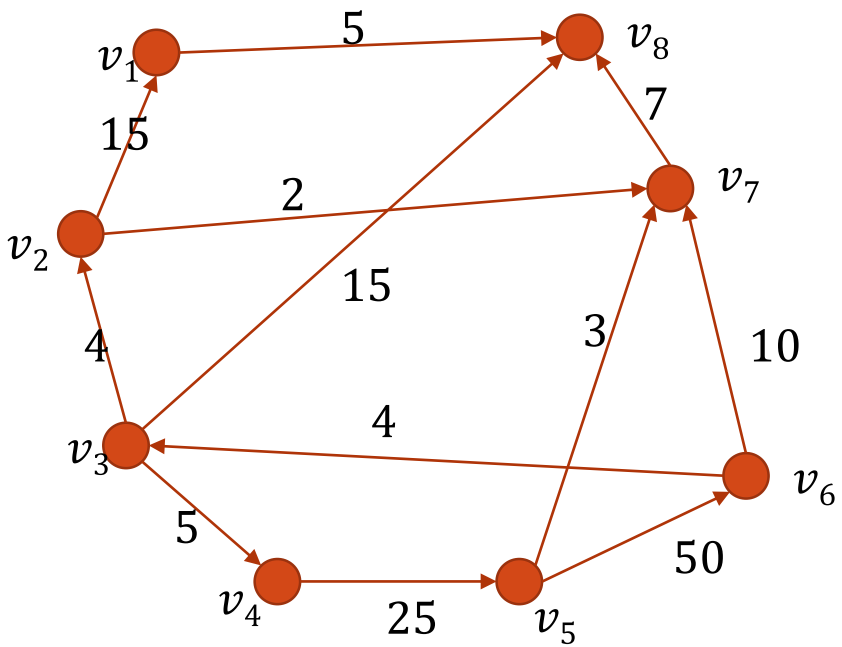
Graphs: Introduction

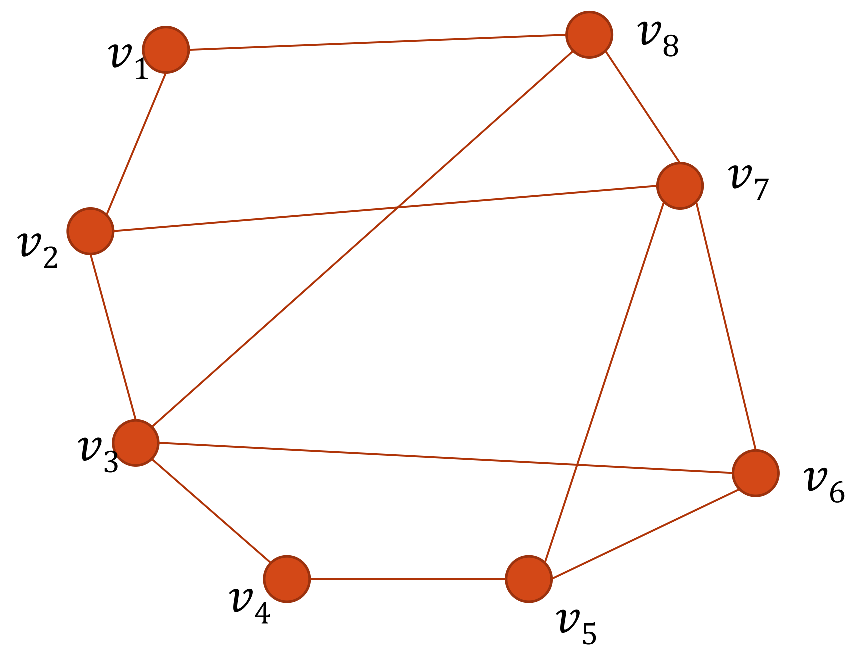
- Path: A sequence of vertices $v_1, v_2, \dots, v_k$ such that for any consecutive pair of vertices v_i, v_{i+1}, (v_i, v_{i+1}) is an edge in the graph. It is called a path from v_1 to v_k. A cycle is a path where $v_1 = v_k$ and $v_1, \dots, v_{k-1}$ are distinct vertices.



Graphs: Introduction

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

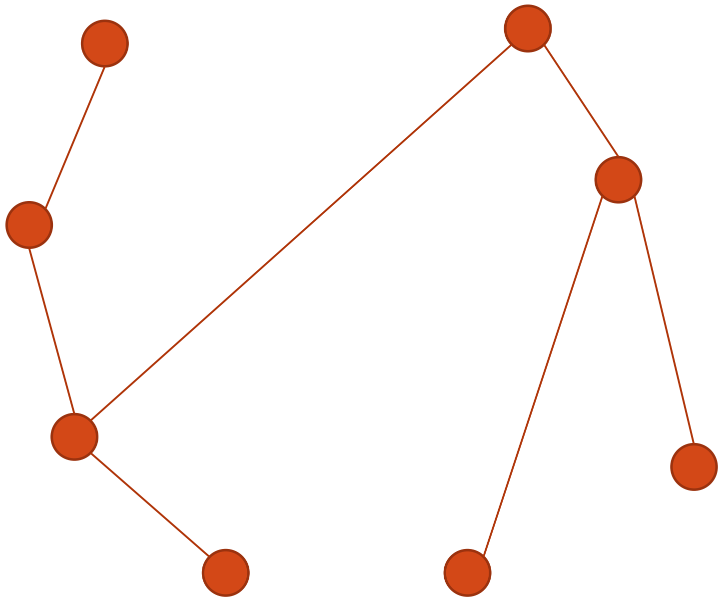




Graphs: Introduction

- <u>Tree</u>: A strongly connected, undirected graph is called a tree if it has no cycles.

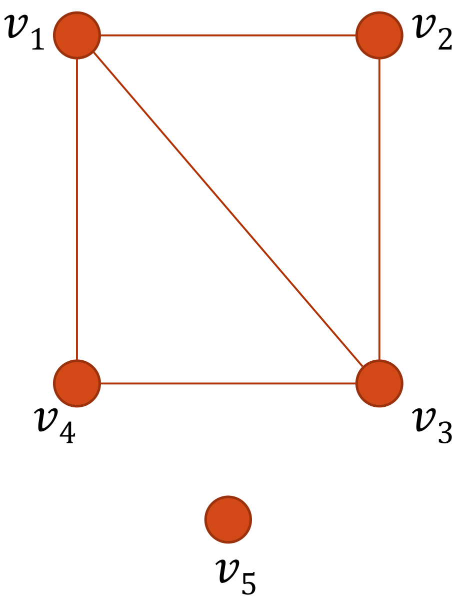
- How many edged does a tree have?



Graph

Data Structures for representing graphs

Graph: Data structures

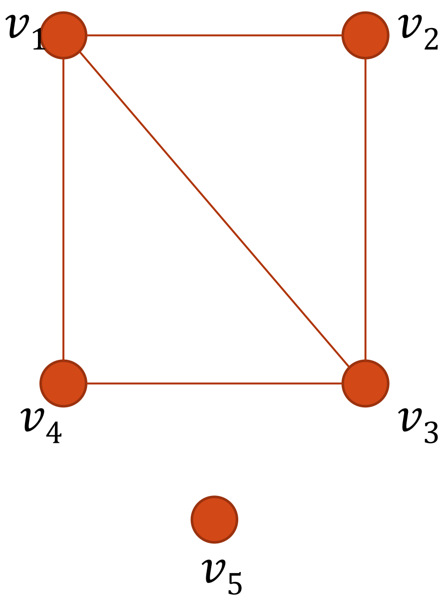


- Adjacency matrix:

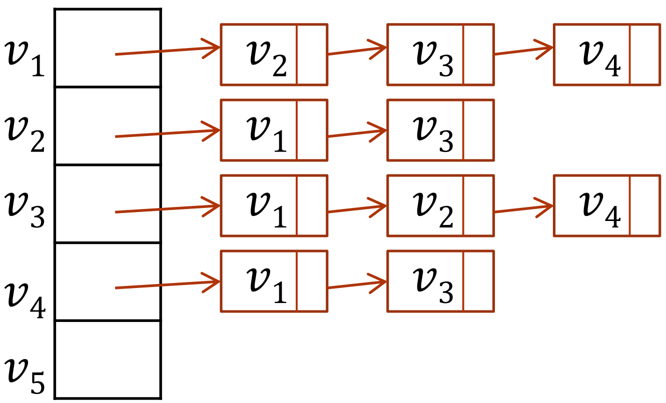
	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	1	0
v_2	1	0	1	0	0
v_3	1	1	0	1	0
v_4	1	0	1	0	0
v_5	0	0	0	0	0

- Space: $O(n^2)$

Graph: Data structures



- Adjacency list: For each vertex store its neighbors



- Space: $O(n + m)$

Graph

Graph algorithms

Graph Algorithms: s-t connectivity

- <u>Problem</u>: Given an (undirected) graph $G = (V, E)$ and two vertices s, t, check if there is a path between s and t.

Graph Algorithms: s-t connectivity

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

- There is a path between s and t iff s and t are in the same connected component.

Graph Algorithms: s-t connectivity

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

- There is a path between s and t iff s and t are in the same connected component.

- Alternate problem: What are the vertices which are reachable from s. Is t among these reachable vertices.
  - Graph exploration: Explore all the vertices reachable from s.

Graph Algorithms: BFS

- Breadth First Search (BFS):

BFS(G, s){
- Layer(0) = $\{s\}$
- $i = 1$
- 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
  - $i = i + 1$

  }

}

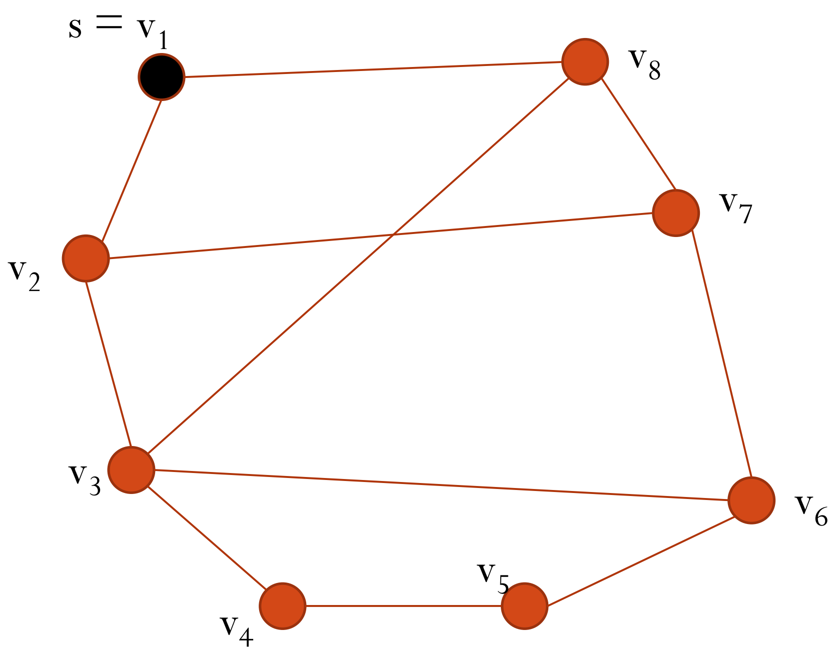
Graph Algorithms: BFS

- Breadth First Search (BFS):

BFS(G, s){
- Layer(0) = {s}
- $i = 1$
- 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
  - $i = i + 1$

  }

}



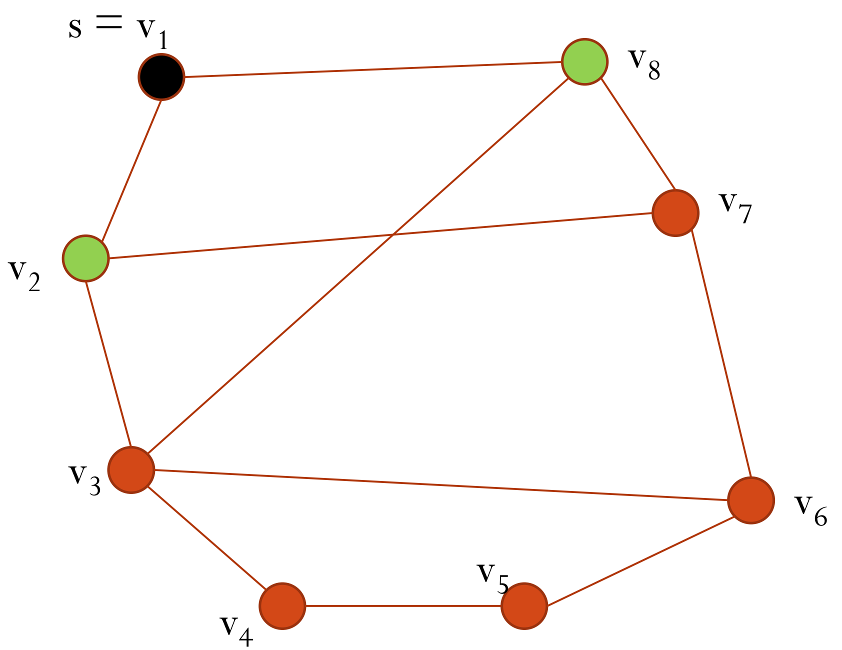
Graph Algorithms: BFS

- Breadth First Search (BFS):

BFS(G, s){
- Layer(0) = {s}
- $i = 1$
- 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
  - $i = i + 1$

  }

}



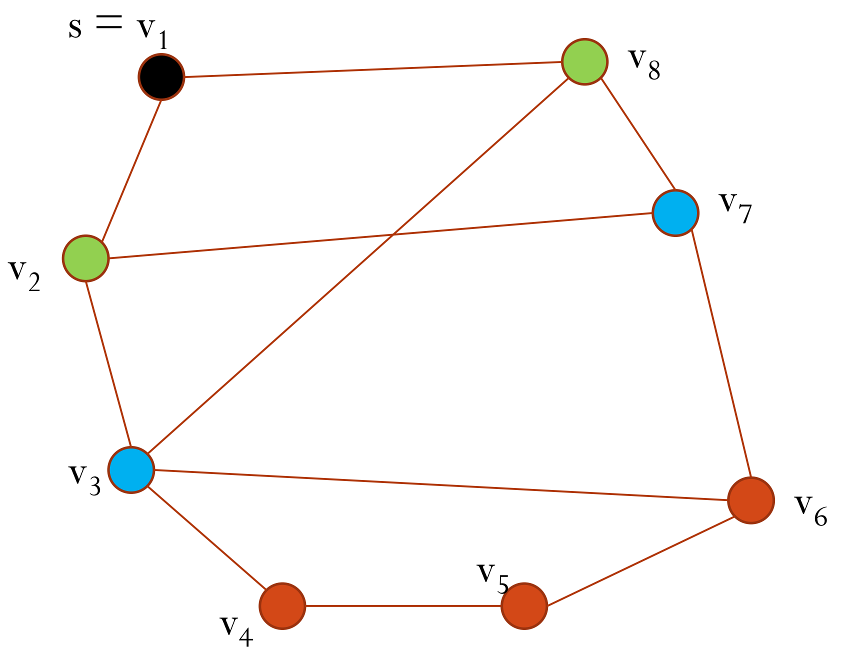
Graph Algorithms: BFS

- Breadth First Search (BFS):

BFS(G, s){
- Layer(0) = {s}
- $i = 1$
- 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
  - $i = i + 1$

  }

}



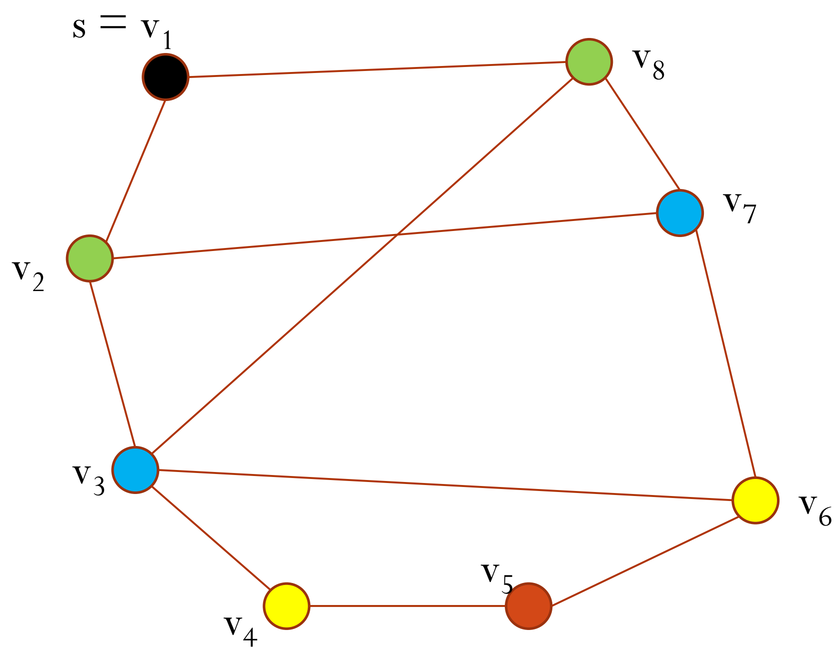
Graph Algorithms: BFS

- Breadth First Search (BFS):

BFS(G, s){
- Layer(0) = {s}
- $i = 1$
- 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
  - $i = i + 1$

  }

}



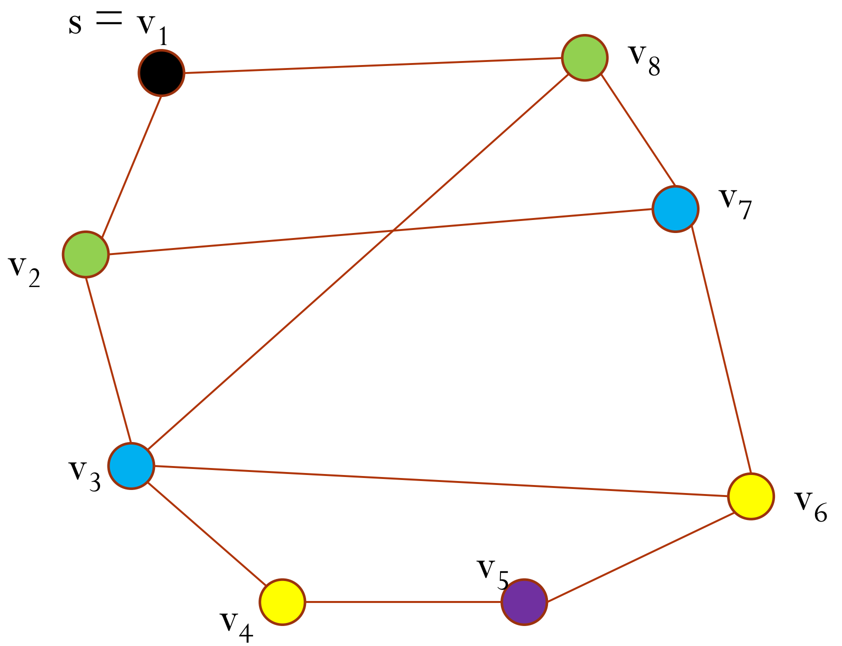
Graph Algorithms: BFS

- Breadth First Search (BFS):

BFS(G, s){
- Layer(0) = {s}
- $i = 1$
- 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
  - $i = i + 1$

  }

}



Show: The shortest path from s to any vertex in $L(i)$ is equal to i.

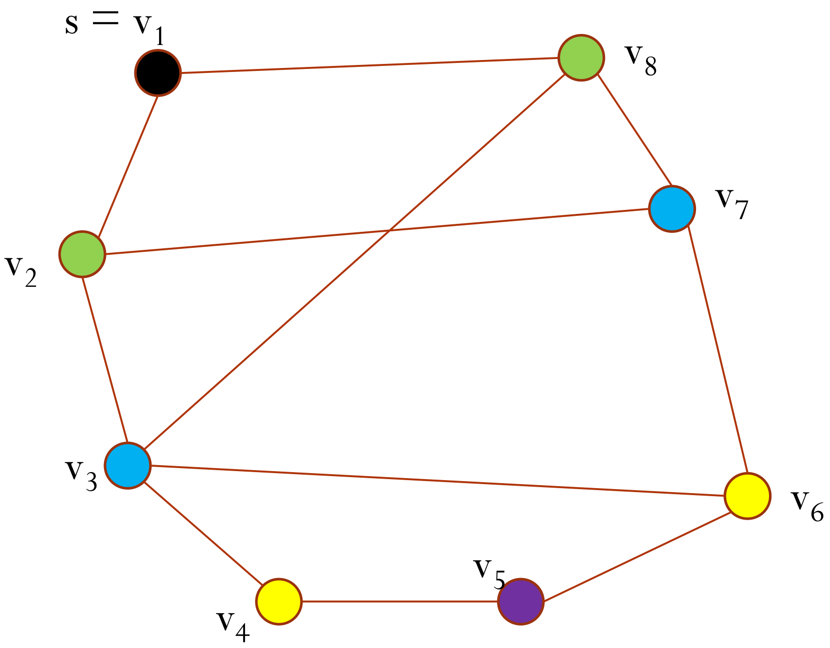
Graph Algorithms: BFS

- Breadth First Search (BFS):

BFS(G, s){
- Layer(0) = {s}
- $i = 1$
- 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
  - $i = i+1$

  }

}



Running time: $O(n+m)$

Graph Algorithms: BFS

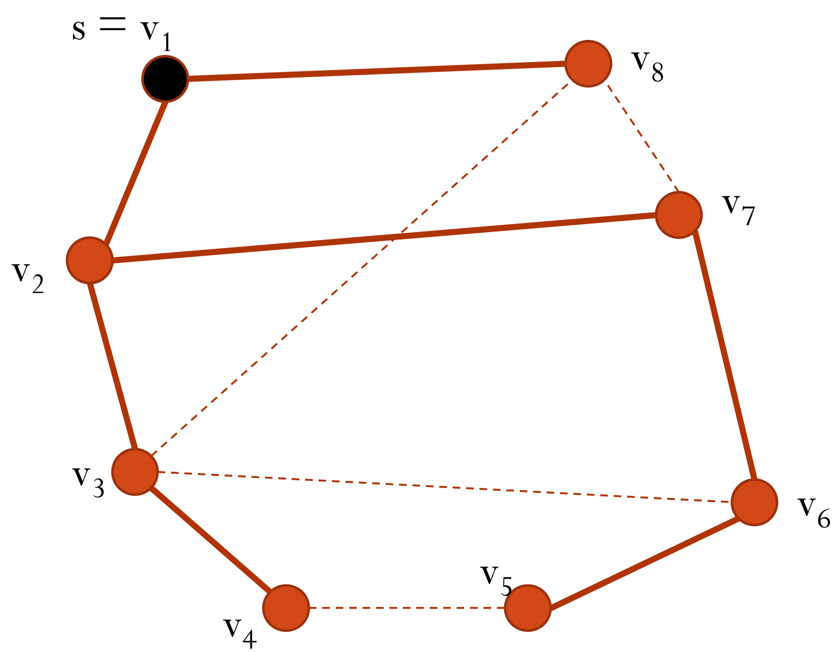
- Breadth First Search (BFS):

```
BFS(G, s){
    - Layer(0) = {s}
    - i = 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 = i + 1
    }
}
```

- The BFS algorithm defines the following "BFS Tree" rooted at s:
  - Vertex u is the parent of vertex v if u caused the immediate discovery of v.

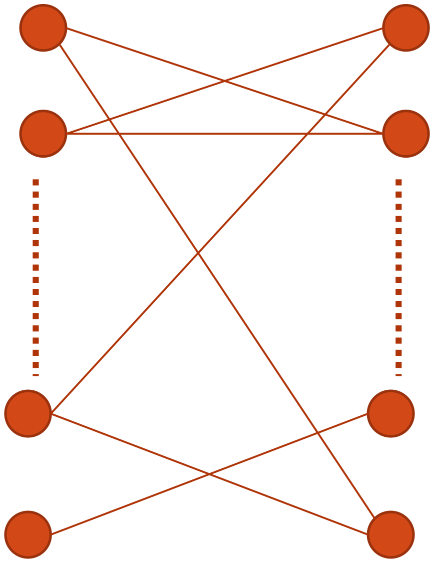
Graph Algorithms: BFS

- Breadth First Search (BFS):
- The BFS algorithm defines the following "BFS Tree" rooted at s:
  - Vertex u is the parent of vertex v if u caused the immediate discovery of v.



Graph Algorithms: BFS

- Problem: Given a graph $G = (V, E)$ check if the graph is *bipartite*.



- A graph is *bipartite* if the vertices can be partitioned into two sets such that there is no edge between any pair of vertices in the same set.

Graph Algorithms: BFS

- Problem: Given a graph $G = (V, E)$ check if the graph is *bipartite*.

```
BFS(G, s){
    - Layer(0) = {s}
    - i = 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 = i + 1
    }
}
```

- Consider BFS:
- Is it possible that there is an edge between vertices which belong to sets $L(i)$ and $L(j)$ such that $(j - i) > 1$?

Graph Algorithms: BFS

- Problem: Given a graph $G = (V, E)$ check if the graph is *bipartite*.

- Suppose the given graph contains a cycle of odd length. Can this graph be bipartite?

Graph Algorithms: BFS

- Problem: Given a graph $G = (V, E)$ check if the graph is *bipartite*.

- Suppose the given graph contains a cycle of odd length. Can this graph be bipartite?

- Can you now use BFS to check if the graph is bipartite?

Graph Algorithms: BFS

- Problem: Given a graph $G = (V, E)$ check if the graph is *bipartite*.

- Suppose the given graph contains a cycle of odd length. Can this graph be bipartite?

- Can you now use BFS to check if the graph is bipartite?

- What is the running time of your algorithm?

Graph Algorithms: BFS

- Problem: Given a graph $G = (V, E)$ check if the graph is *bipartite*.

- Suppose the given graph contains a cycle of odd length. Can this graph be bipartite?

- Can you now use BFS to check if the graph is bipartite?

- What is the running time of your algorithm?

- Suppose a graph does not have an odd cycle. Does that mean that the graph is bipartite?

End

Problems to think about:

1. The BFS algorithm gives the shortest path from s to any vertex. Suppose we are given a weighted graph where the weights are numbers between 1 and 10. Can you use the BFS algorithm to find the shortest length path from s to any vertex in the graph. The length of a path is the sum of weights of edges in the path.