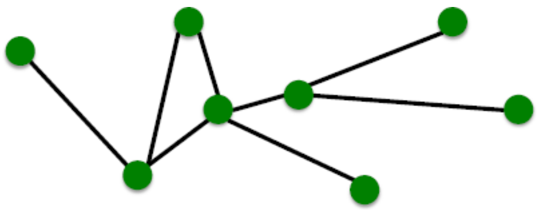
Graphs

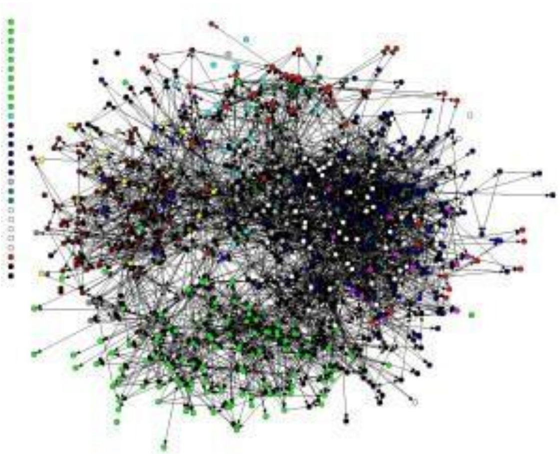
- Depth First Search (DFS)
- Breath First Search (BFS)

Introduction

- **Graph – A tool to model binary relationships among entities/objects Specified by two sets – Set of vertices V and set of Edges E**
- **Vertex/Node – represent entities**
- **Edge – represent existence of relationship between a pair of entities A graph G(V, E) where $E \subseteq VXV$**

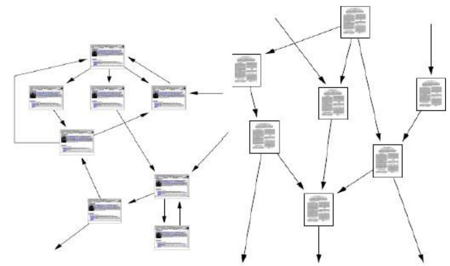


Graph everywhere



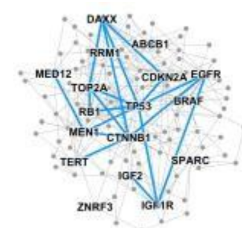
Email Communication Network

Social networks



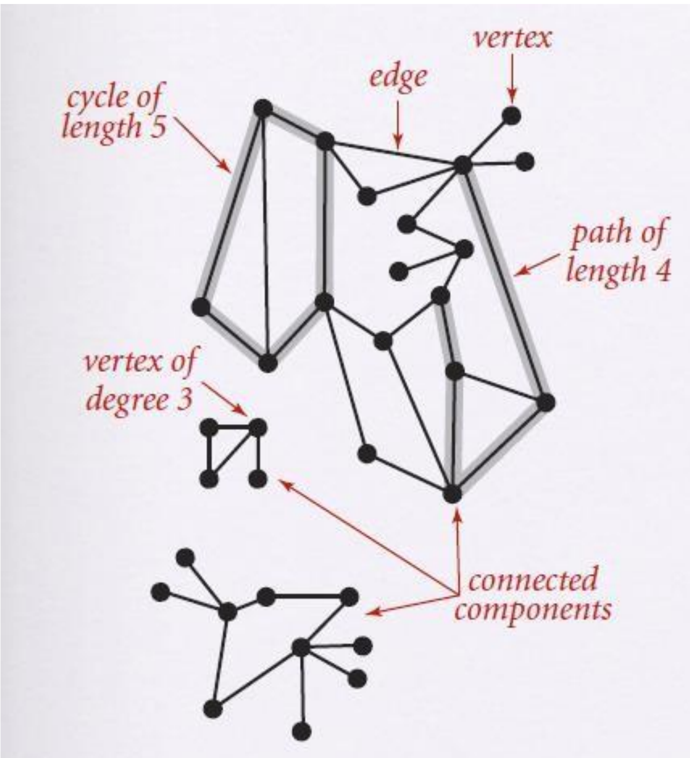
Information networks:
Web & citations

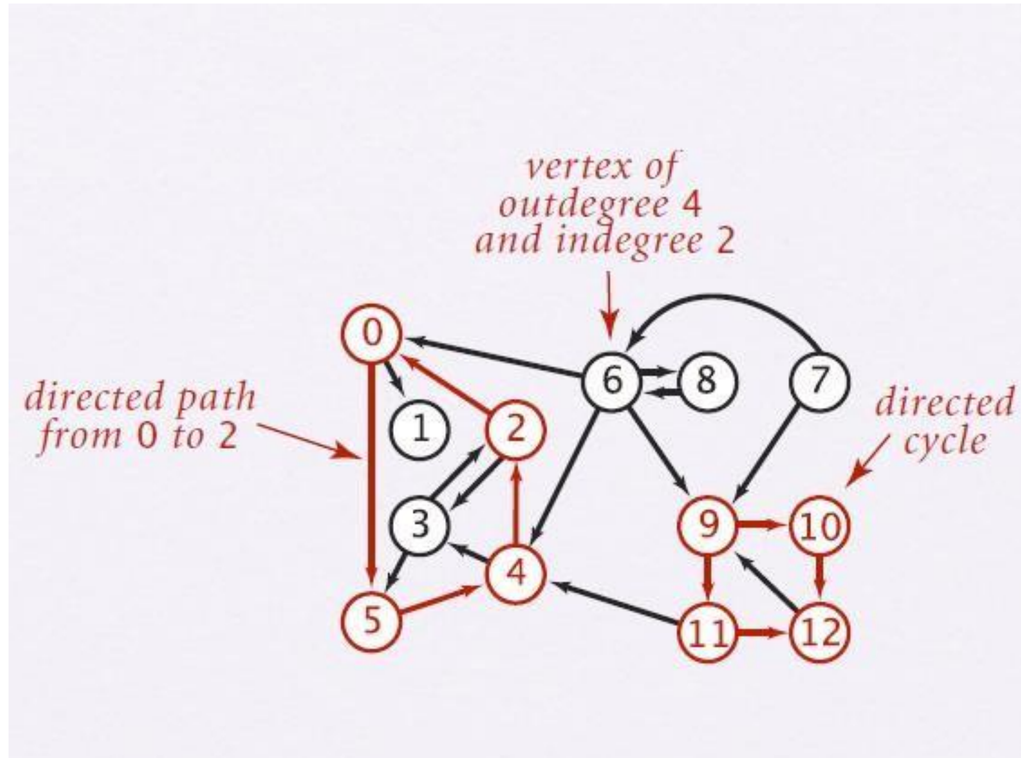
Patient networks



Disease pathways

Graph terminology





Undirected graph

Directed graph

Handshaking Lemma

- **Undirected Graph**

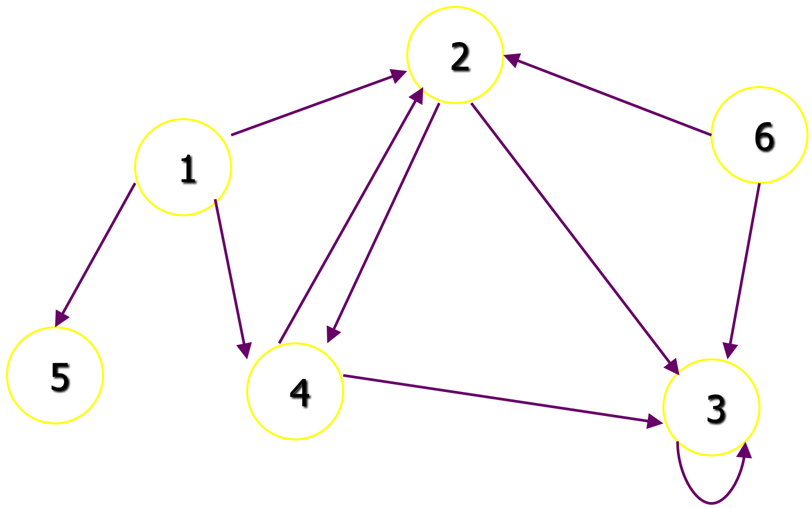
  Lemma: Sum of degrees of all the vertices is twice the number of edges

  $$\sum_{v \in V} \deg(v) = 2|E|$$

- **Directed Graph**

  Lemma: Sum of in-degrees of all vertices is same as sum of out-degrees of all vertices is same as total number of edges

  $$\sum_{v \in V} \mathrm{Indeg}(v) = \sum_{v \in V} \mathrm{Outdeg}(v) = |E|$$

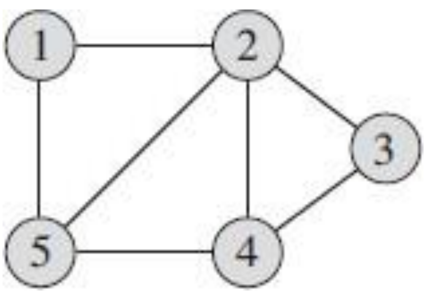

2
1
6
5
4
3

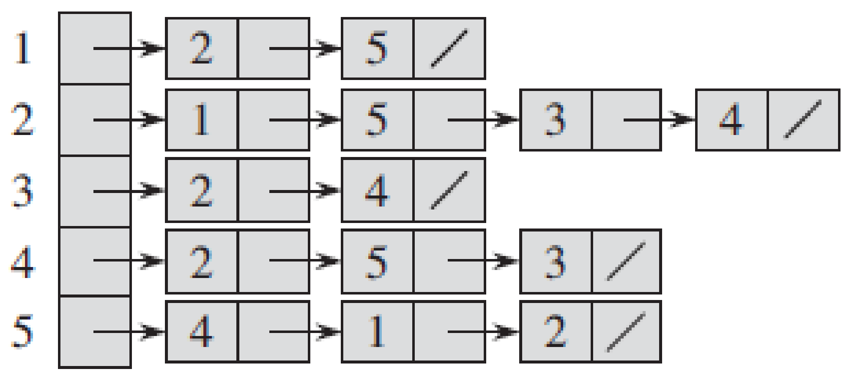
- Trees are special kinds of directed graphs and are characterized by the fact that one of their nodes, the root , has no incoming arcs and every other node can be reached from the root by a unique path, i.e., by following one and only one sequence of consecutive arcs.

- In the preceding digraph, vertex 1 is “rootlike” node having no incoming arcs, but there are many different paths form vertex 1 to various other nodes. So that is not tree. For example , to vertex 3.

Adjacency Matrix v.s. Adjacency List

G=<V, E>





	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Undirected: $|V| + 2|E|$

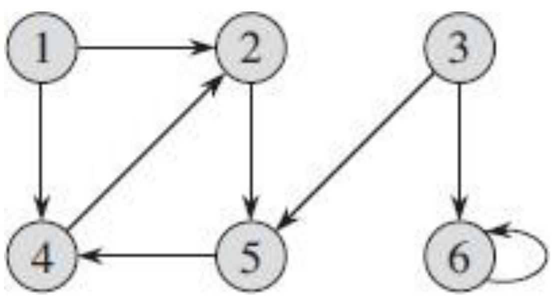
Undirected: $|V|^2$

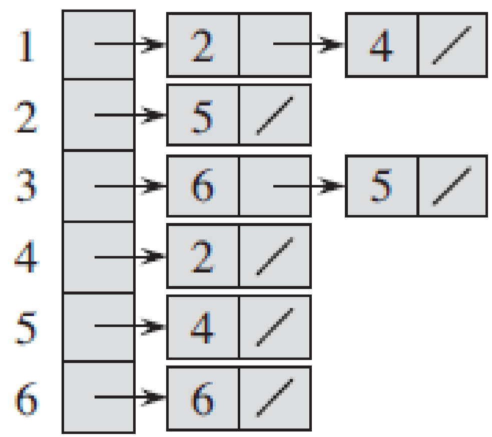
For every edge connected with *v* ...

Is *u* and *v* connected with an edge?

Adjacency Matrix v.s. Adjacency List

G=<V, E>





	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Directed: $|V| + |E|$

Directed: $|V|^2$

For every outgoing edge connected with *v* …

Is *u* has an outgoing edge with v?

Analysis

- For both directed/undirecdted graphs, the adjacency list representation has the desirable property that the amount of memory it requires θ(V+E)
- A potential disadvantage of the adjacency list representation is that there is no quicker way to determine if a given edge (u,v) is present in the graph than to search for v in the adjacency list Adj[u]. This disadvantage can be remedied by an adjacency matrix representation at the cost of using asymptotically more memory.

Analysis

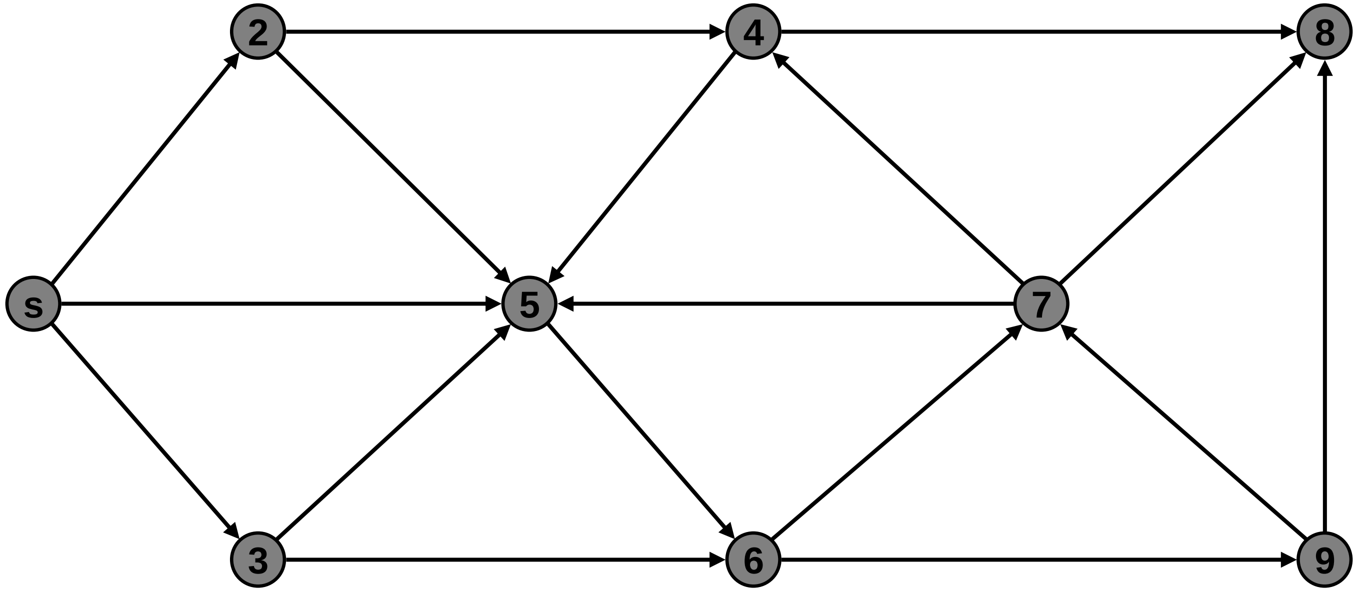
- Adjacency matrix of a graph requires $\theta(V^2)$ memory, independent of the number of edges in the graph.

- Although the adjacency list representation is asymptotically at least as efficient as the adjacency matrix representation , the simplicity of adjacency matrix may make it preferable when graphs are reasonably small.

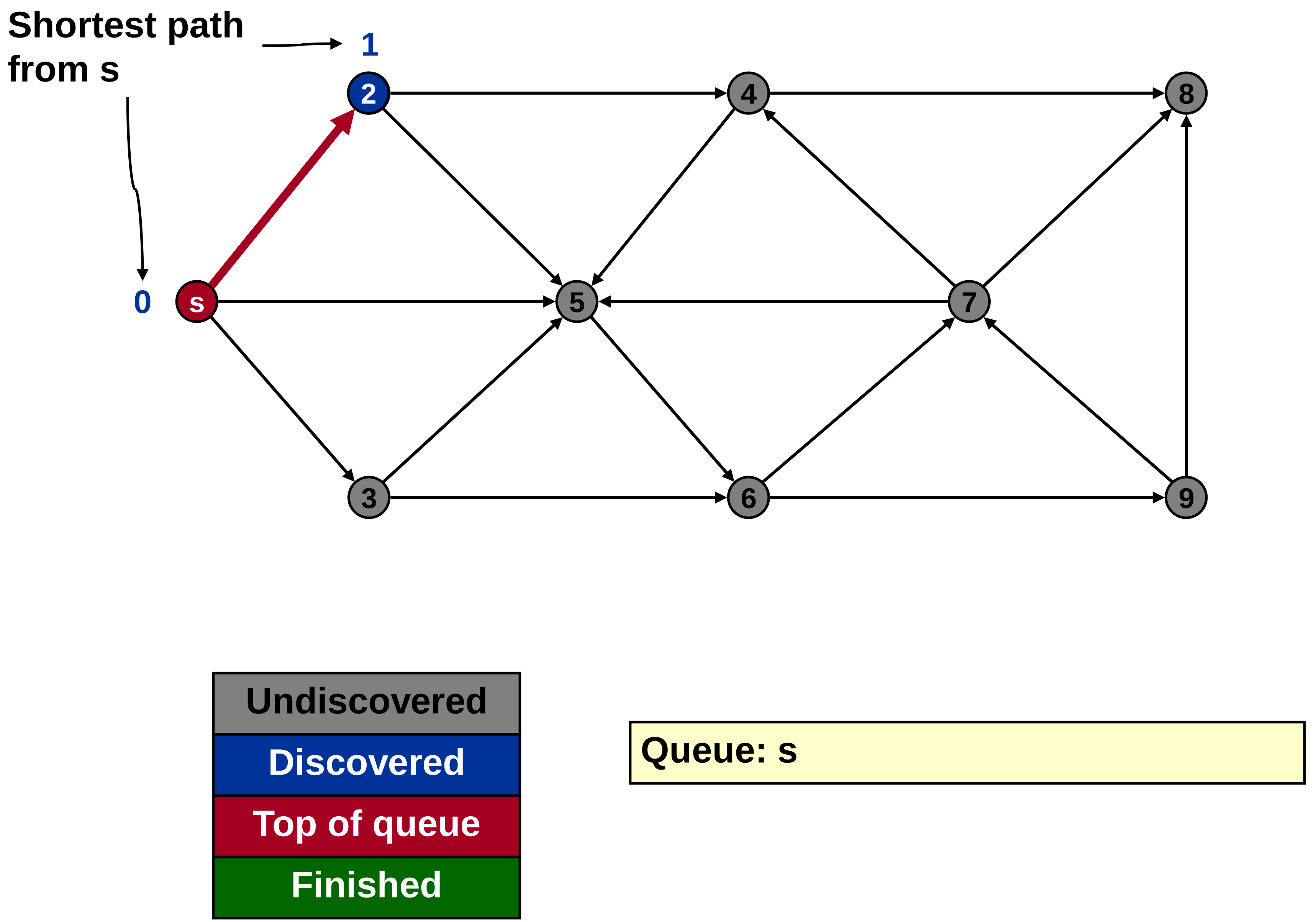
Traversing Graphs

- Two common types of graph traversals are Depth First Search (DFS) and Breadth First Search (BFS).
- DFS is implemented with a stack, and BFS with a queue.
- The aim in both types of traversals is to visit each vertex of a graph *exactly once*.
- In DFS, you follow a path as far as you can go before backing up. With BFS, you visit all the neighbors of the current node before exploring further a nodes in the graph.

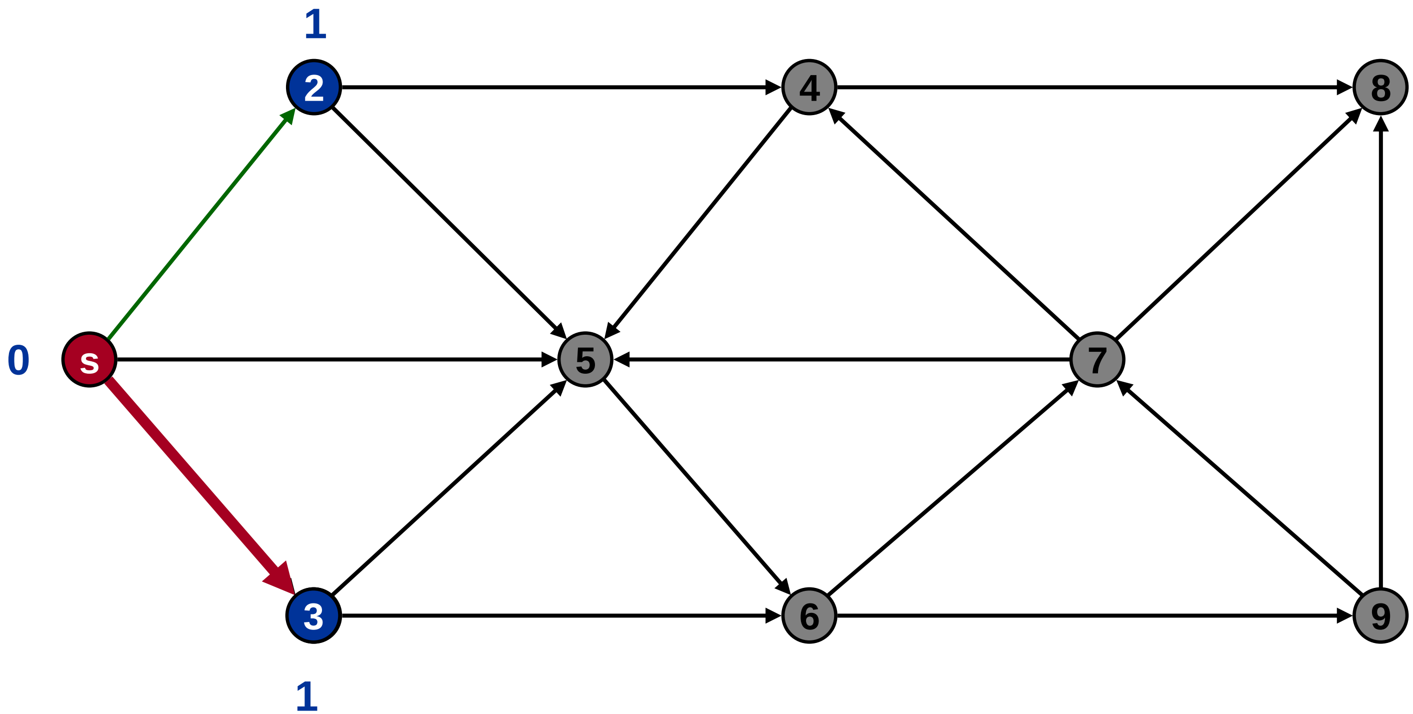
Breadth First Search



Breadth First Search



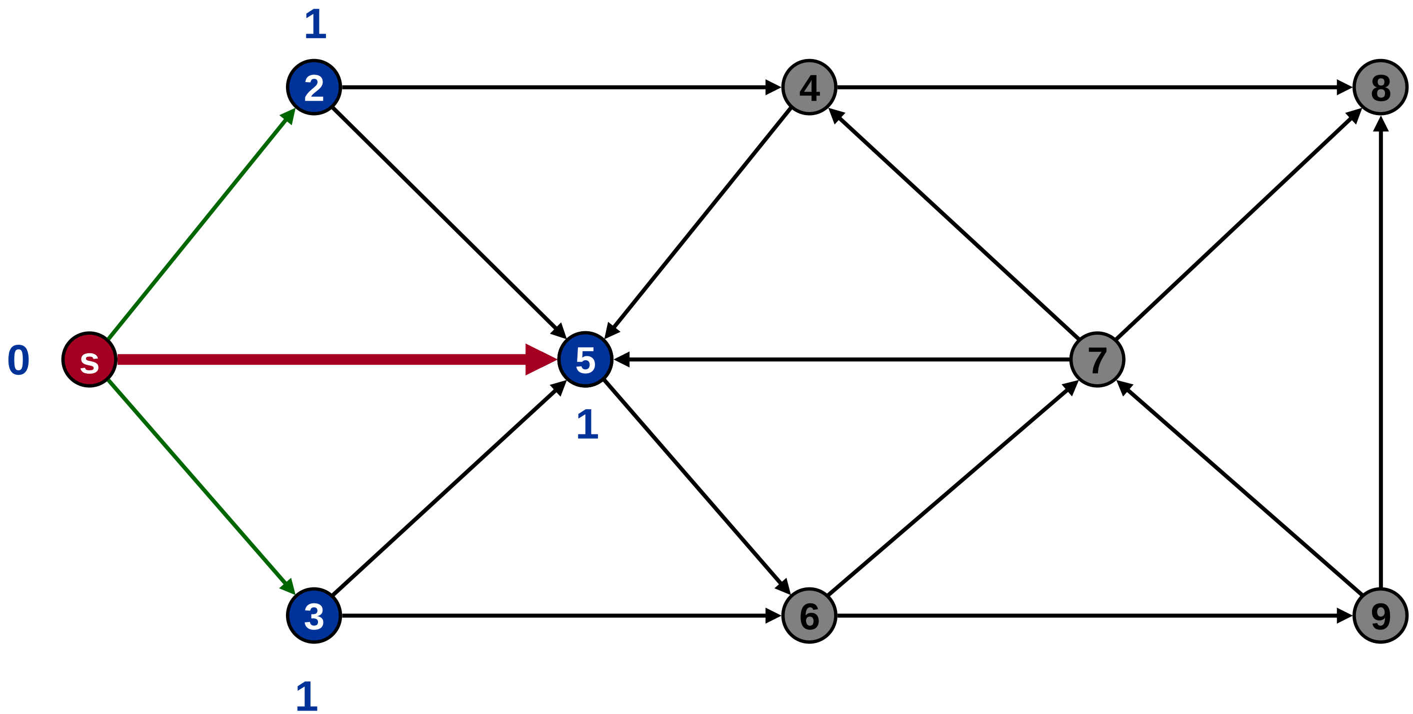
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

Queue: s 2

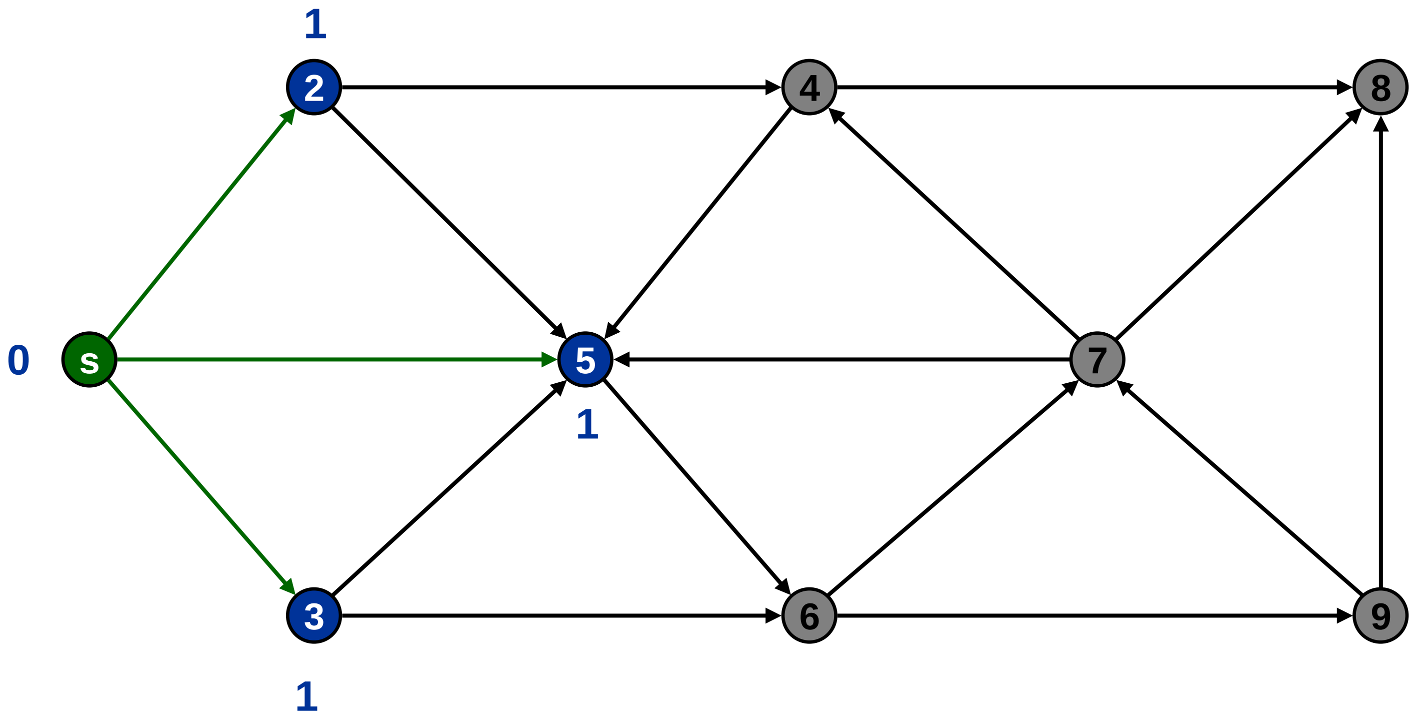
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

Queue: s 2 3

Breadth First Search



Undiscovered

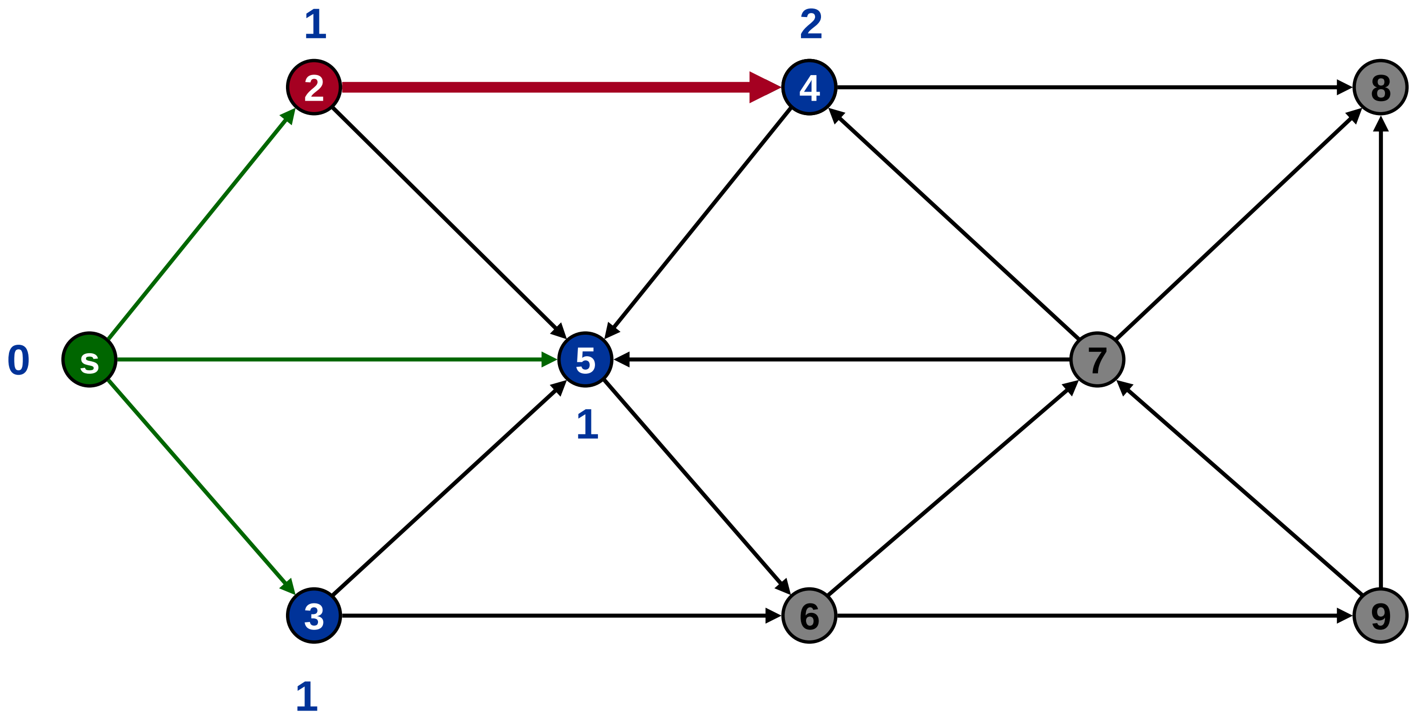
Discovered

Top of queue

Finished

Queue: 2 3 5

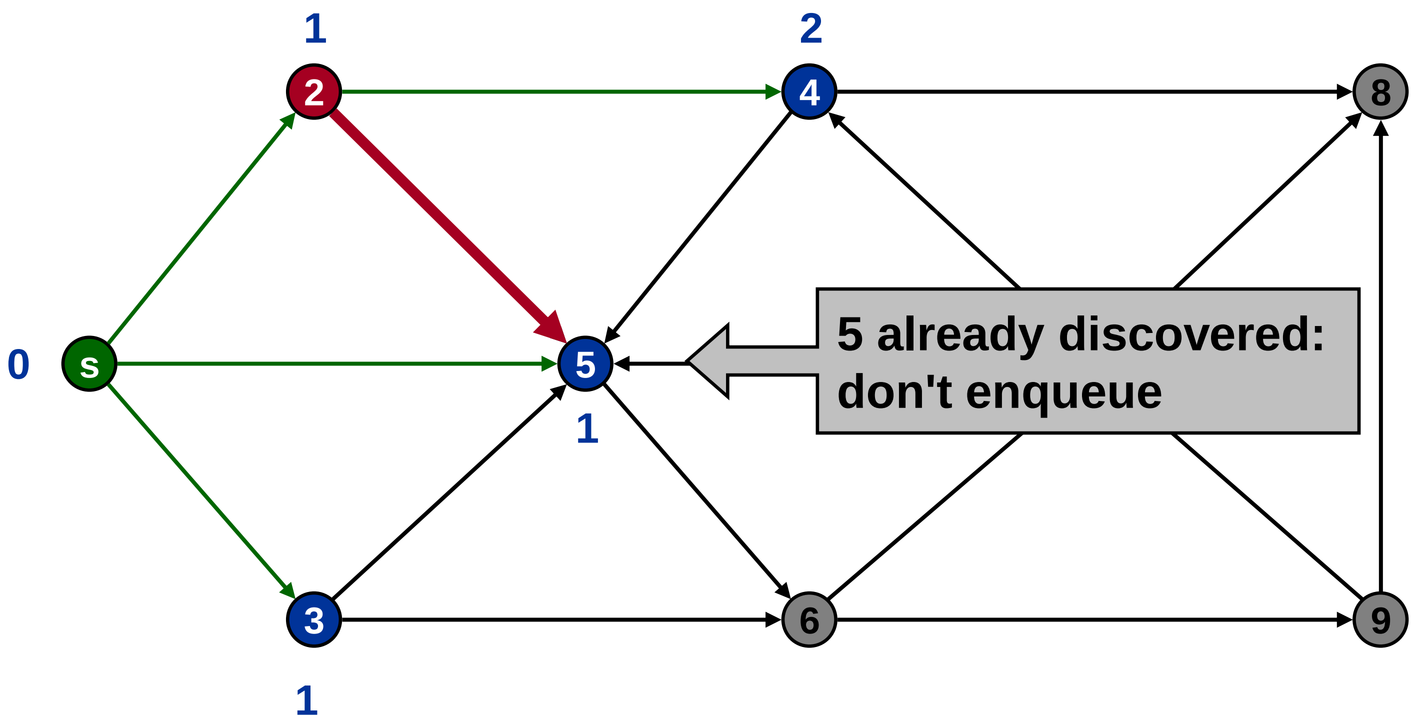
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

Queue: 2 3 5

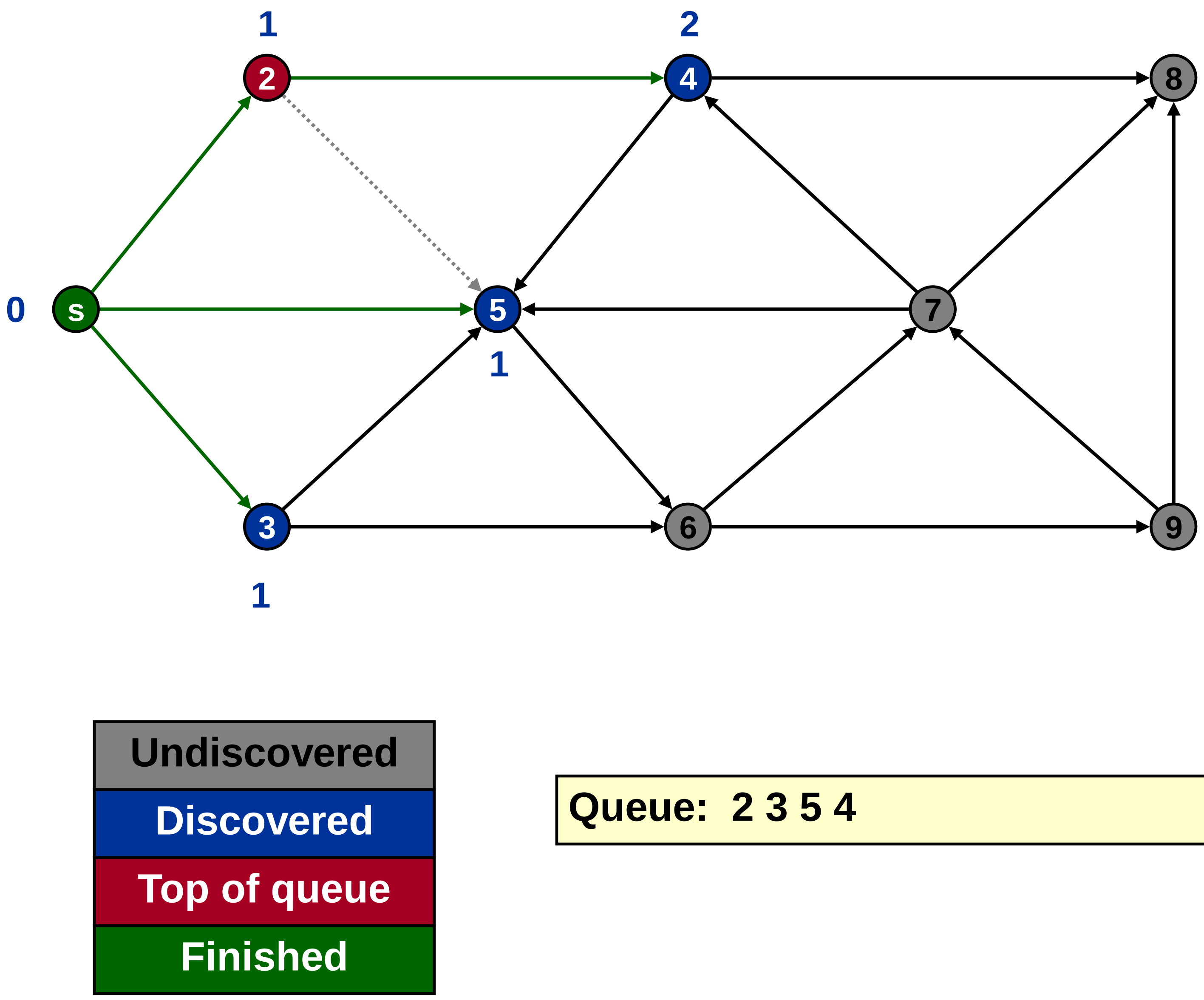
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

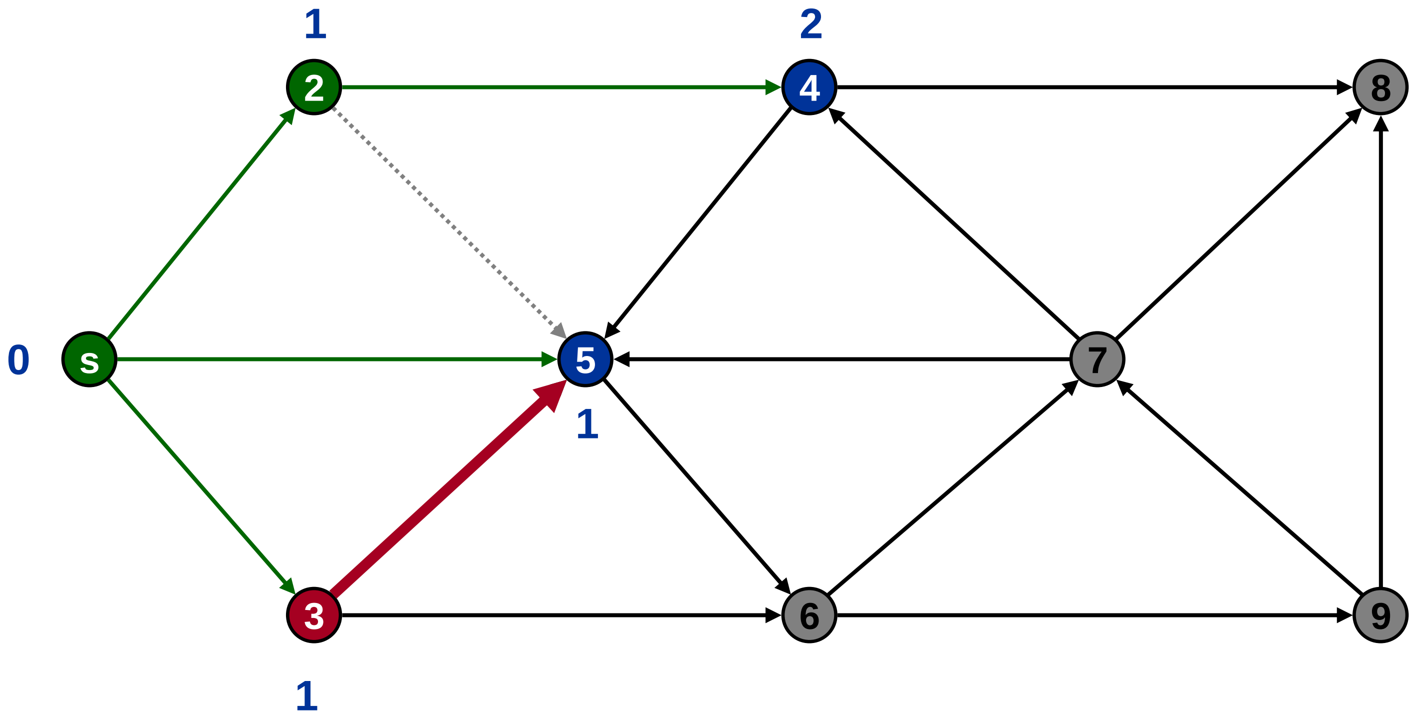
Queue: 2 3 5 4

Breadth First Search



Queue: 2 3 5 4

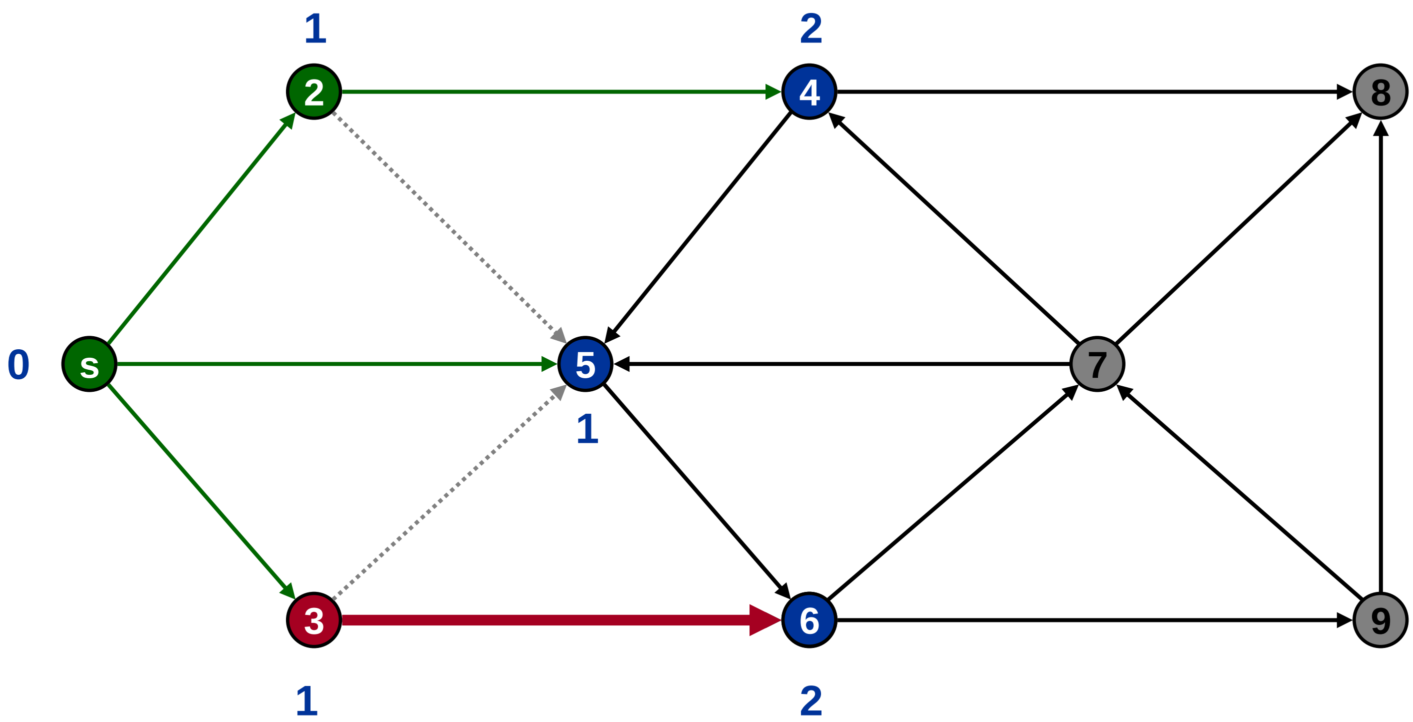
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

Queue: 3 5 4

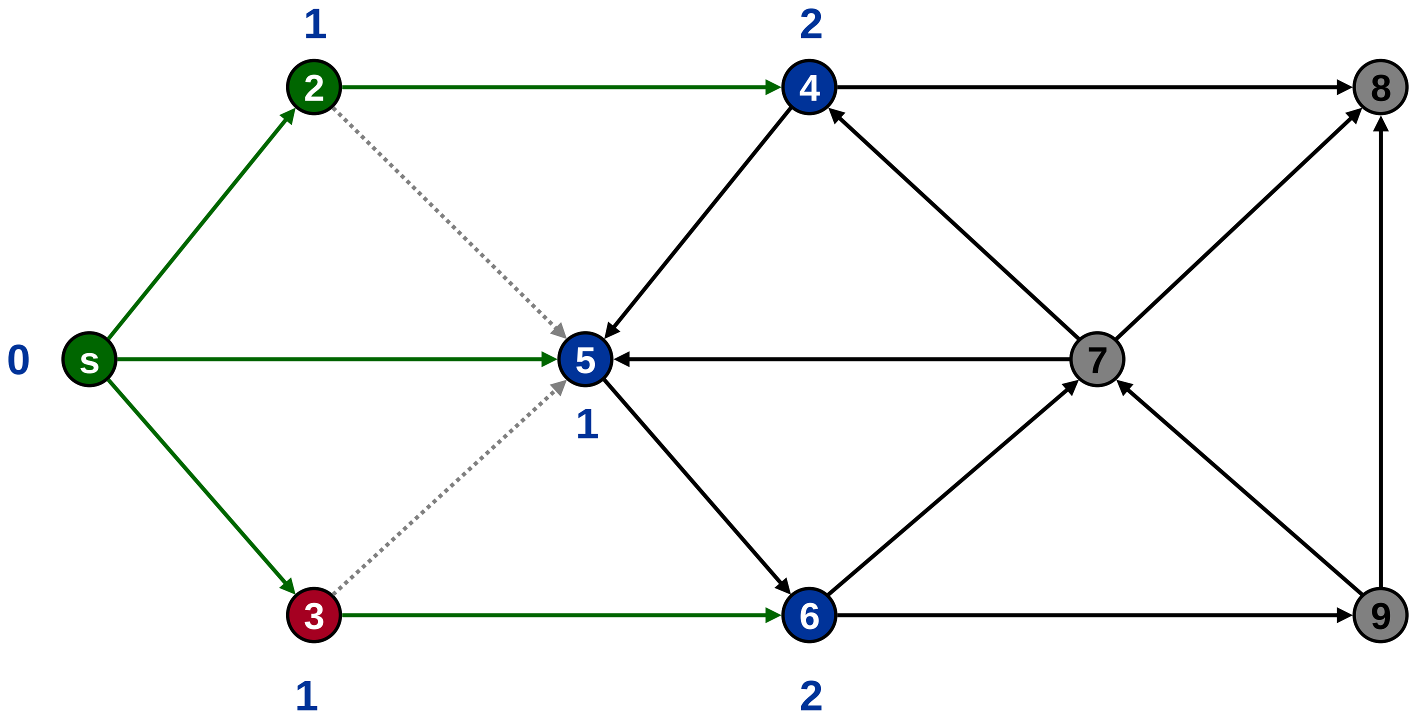
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

Queue: 3 5 4

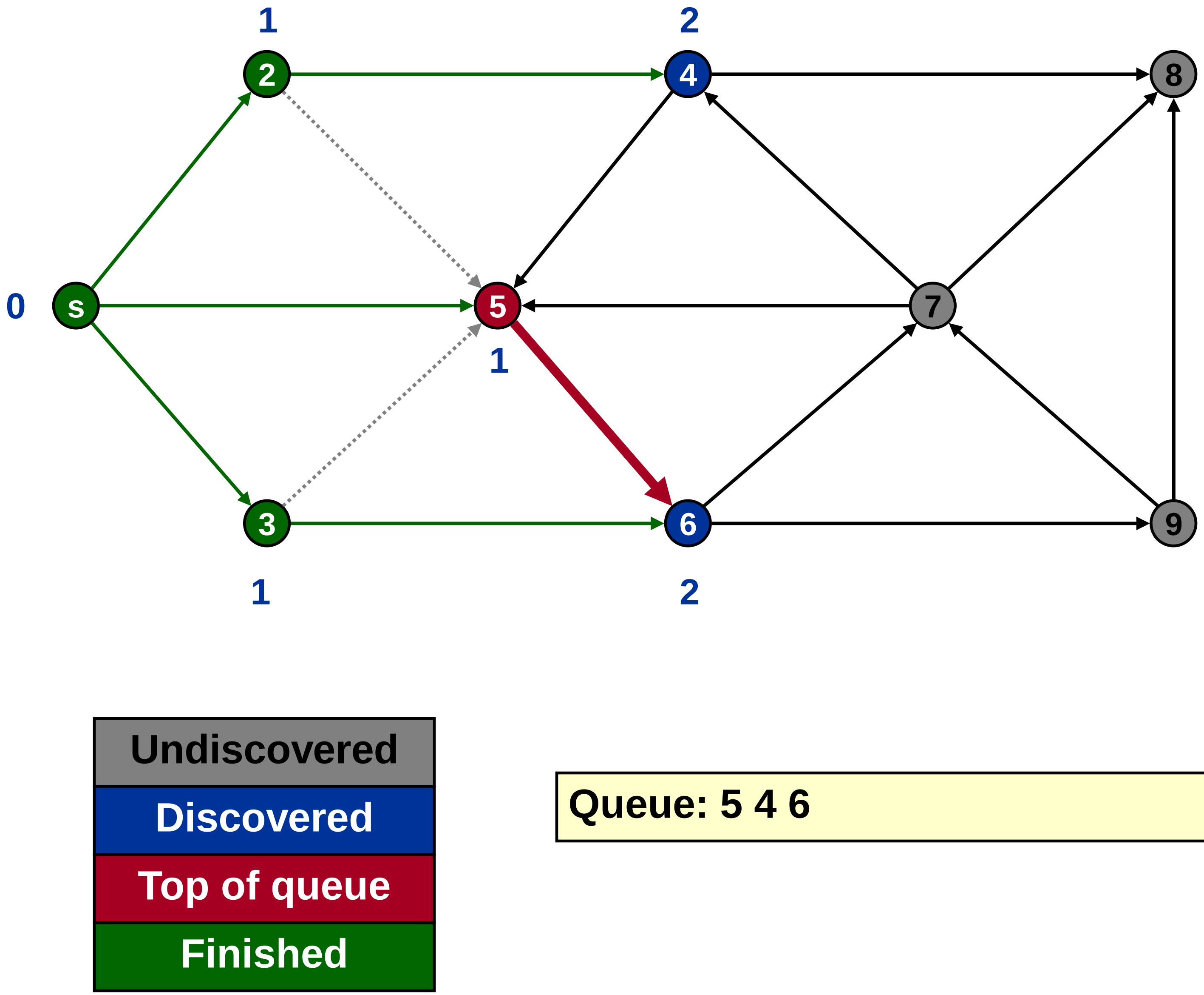
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

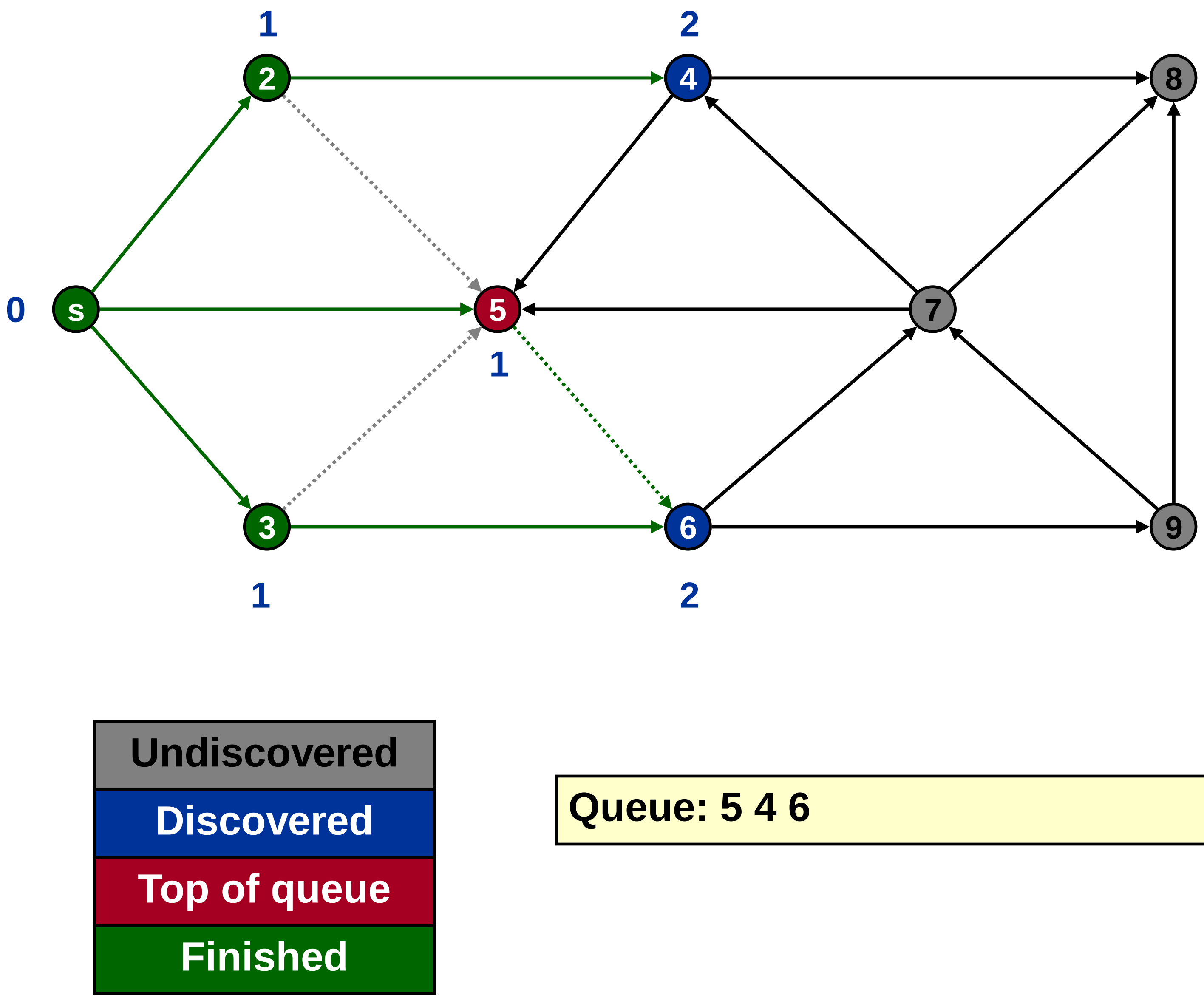
Queue: 3 5 4 6

Breadth First Search



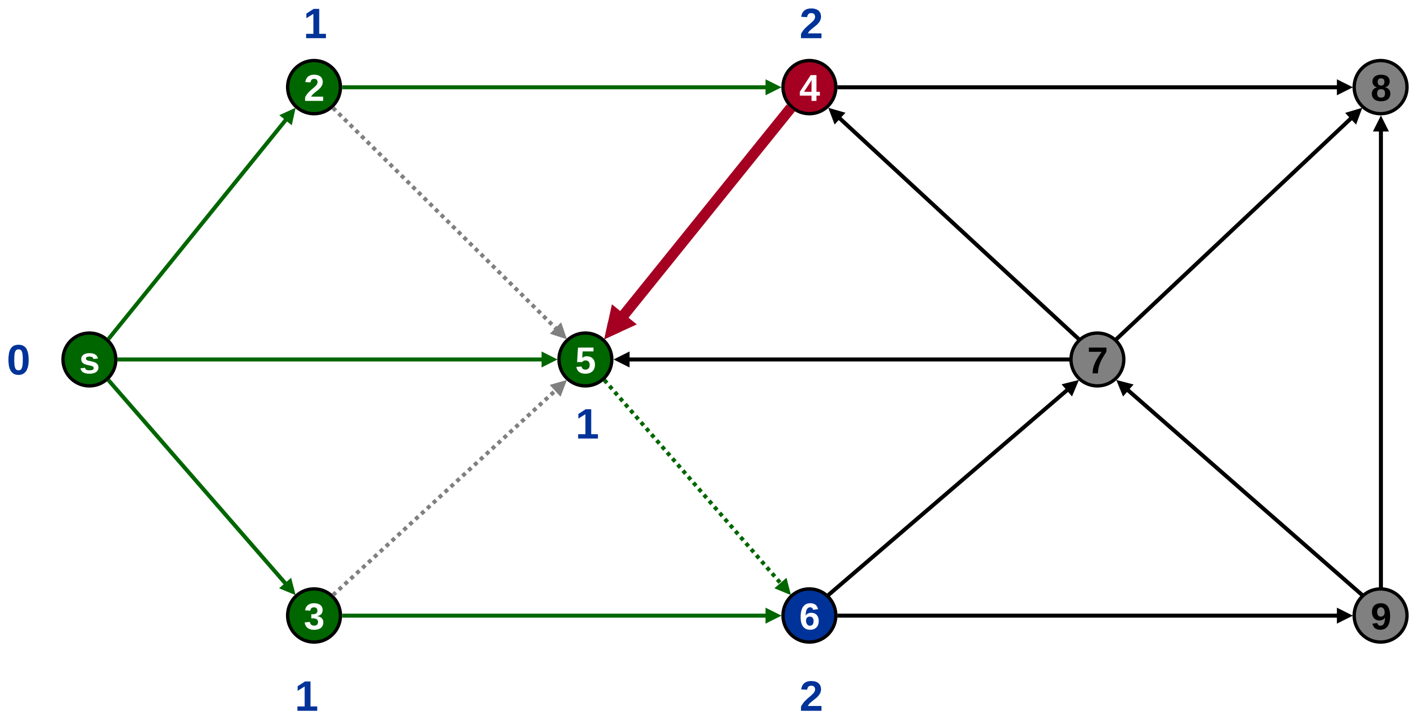
Queue: 5 4 6

Breadth First Search



Queue: 5 4 6

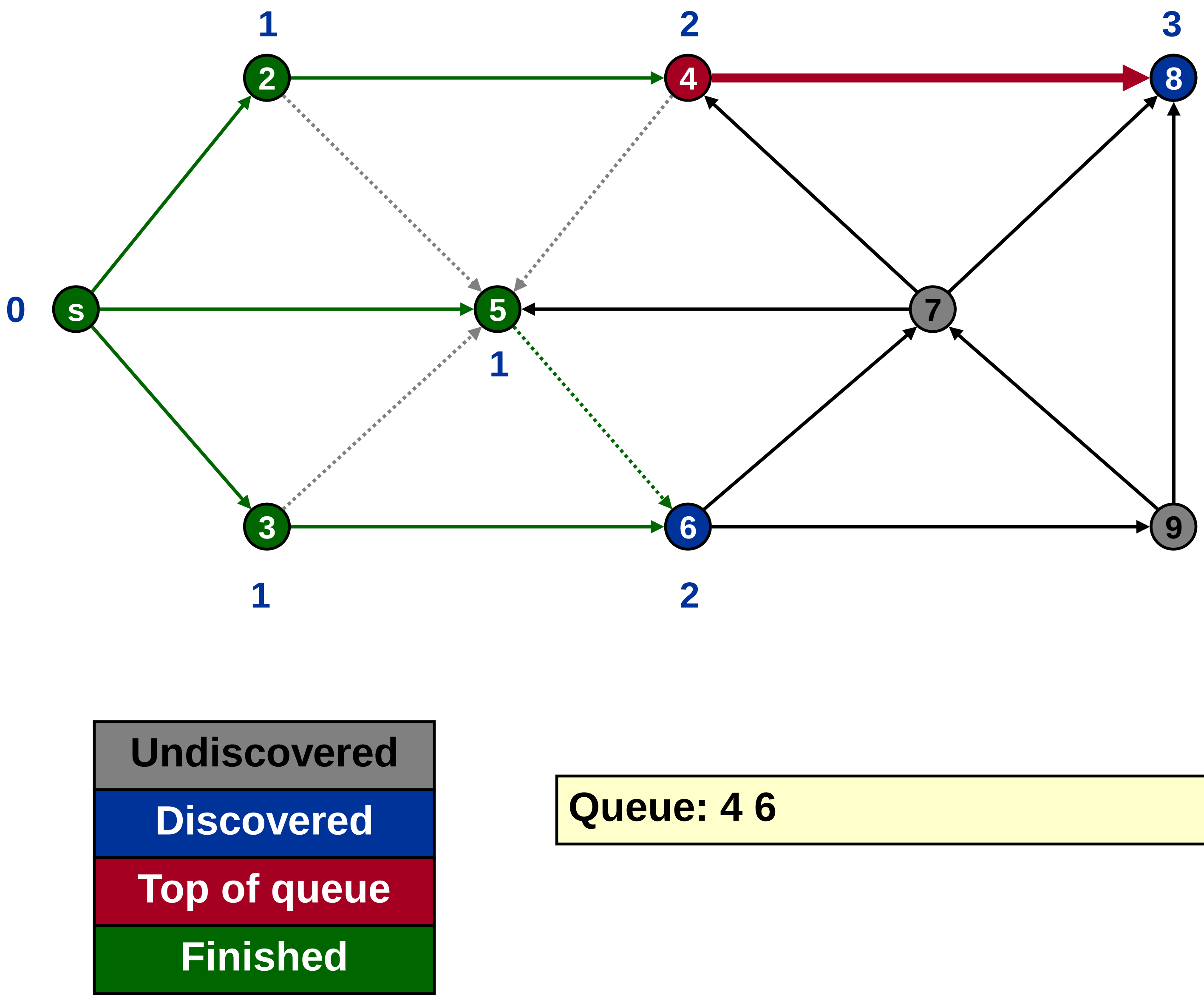
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

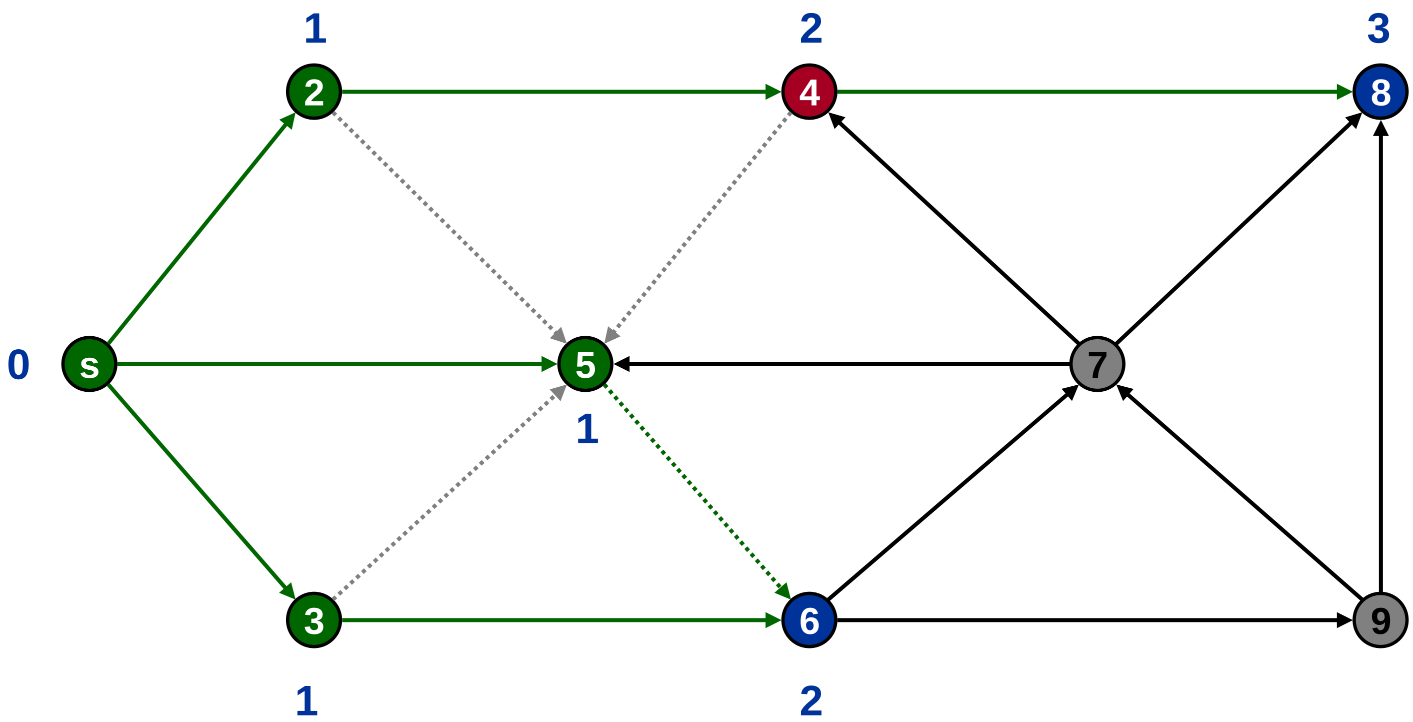
Queue: 4 6

Breadth First Search



Queue: 4 6

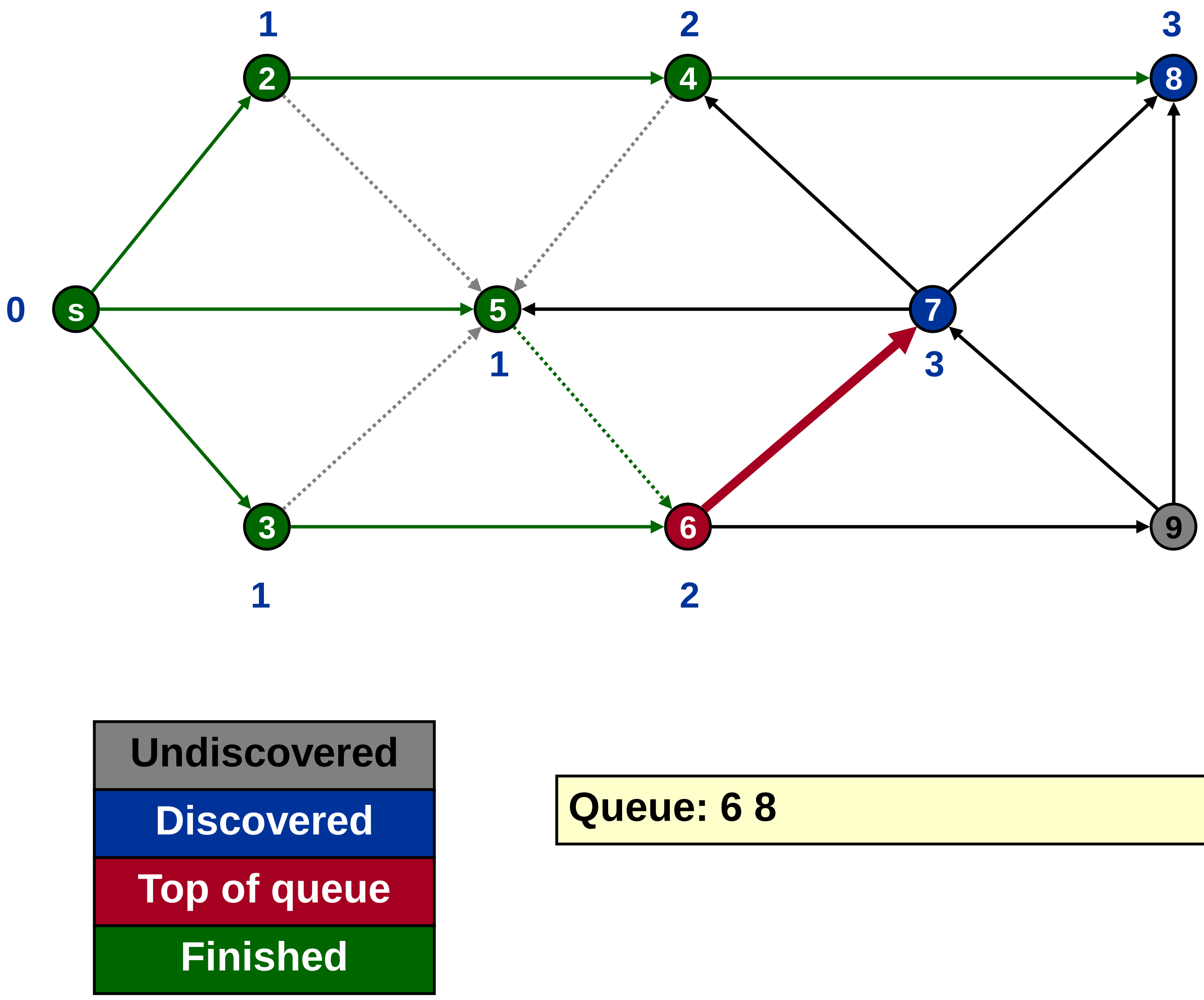
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

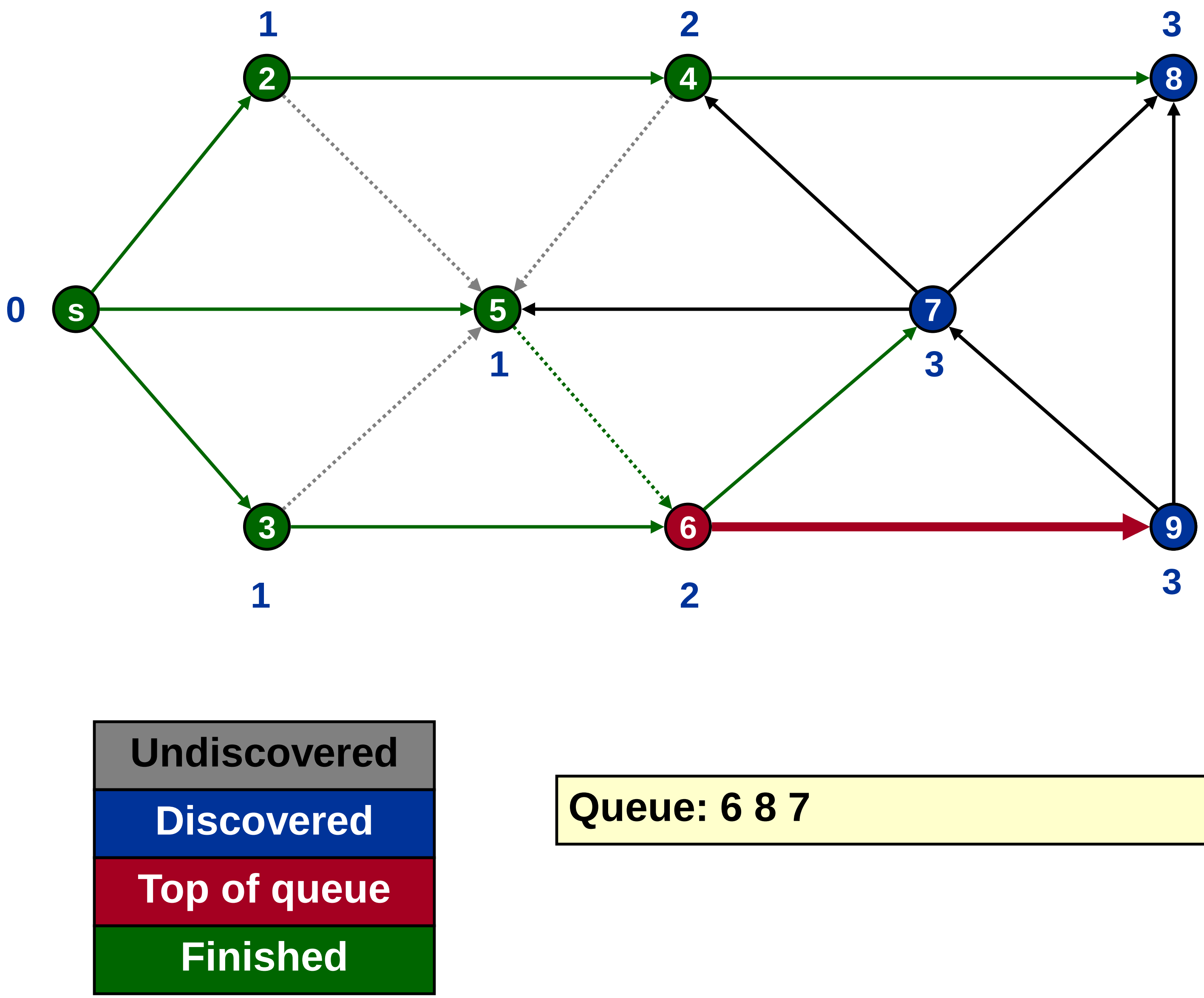
Queue: 4 6 8

Breadth First Search



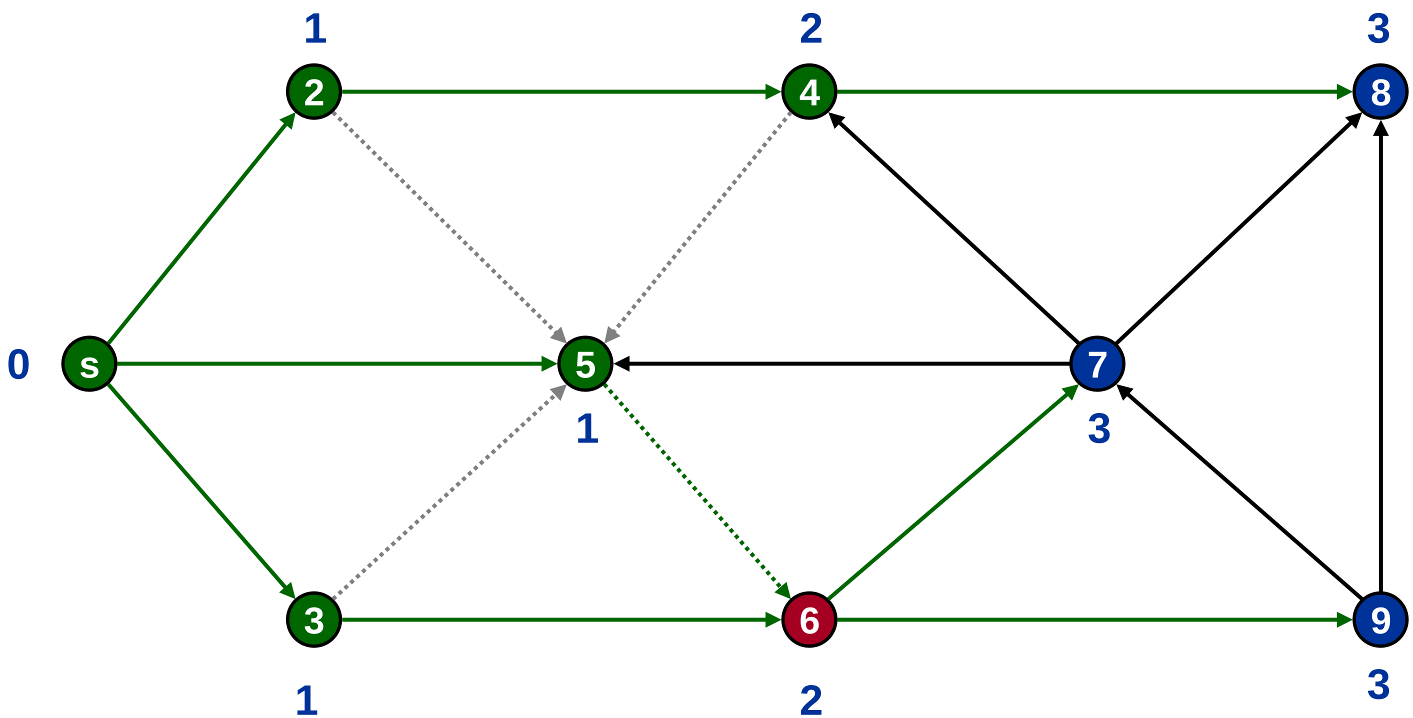
Queue: 6 8

Breadth First Search



Queue: 6 8 7

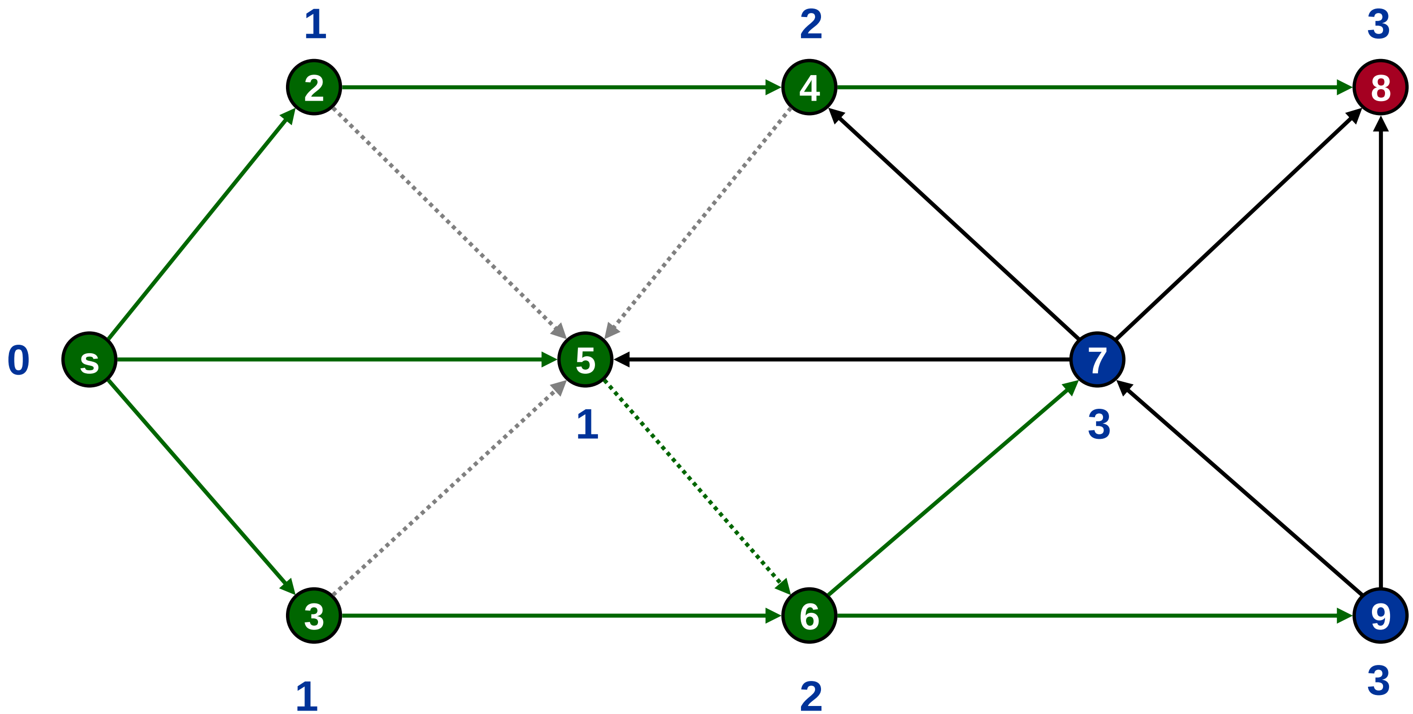
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

Queue: 6 8 7 9

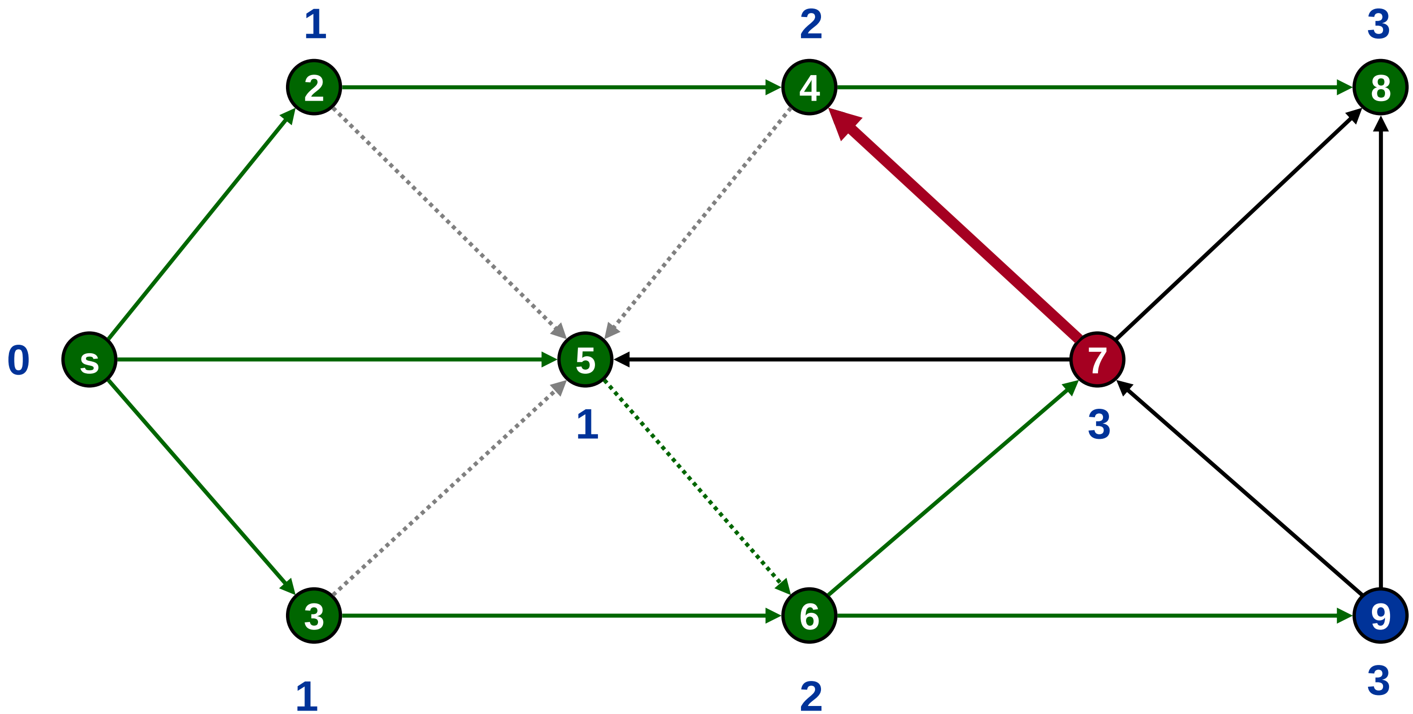
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

Queue: 8 7 9

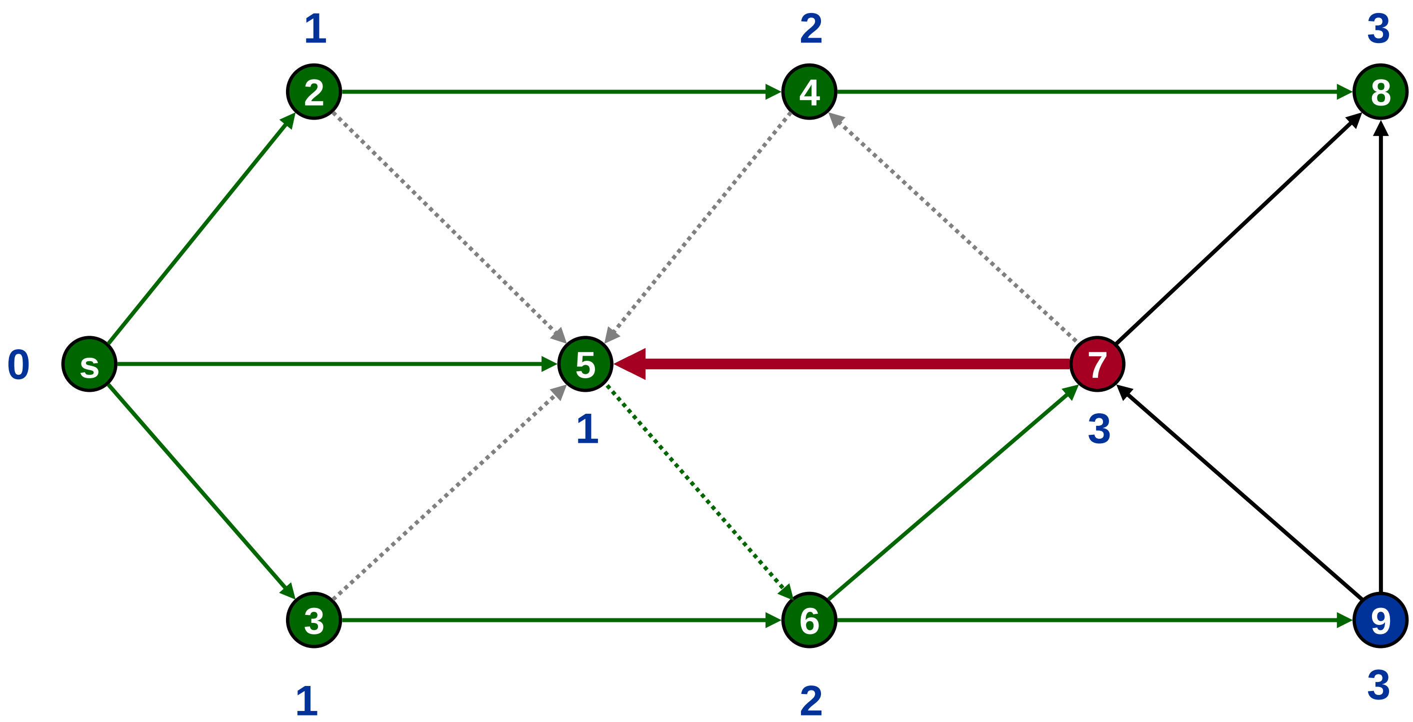
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

Queue: 7 9

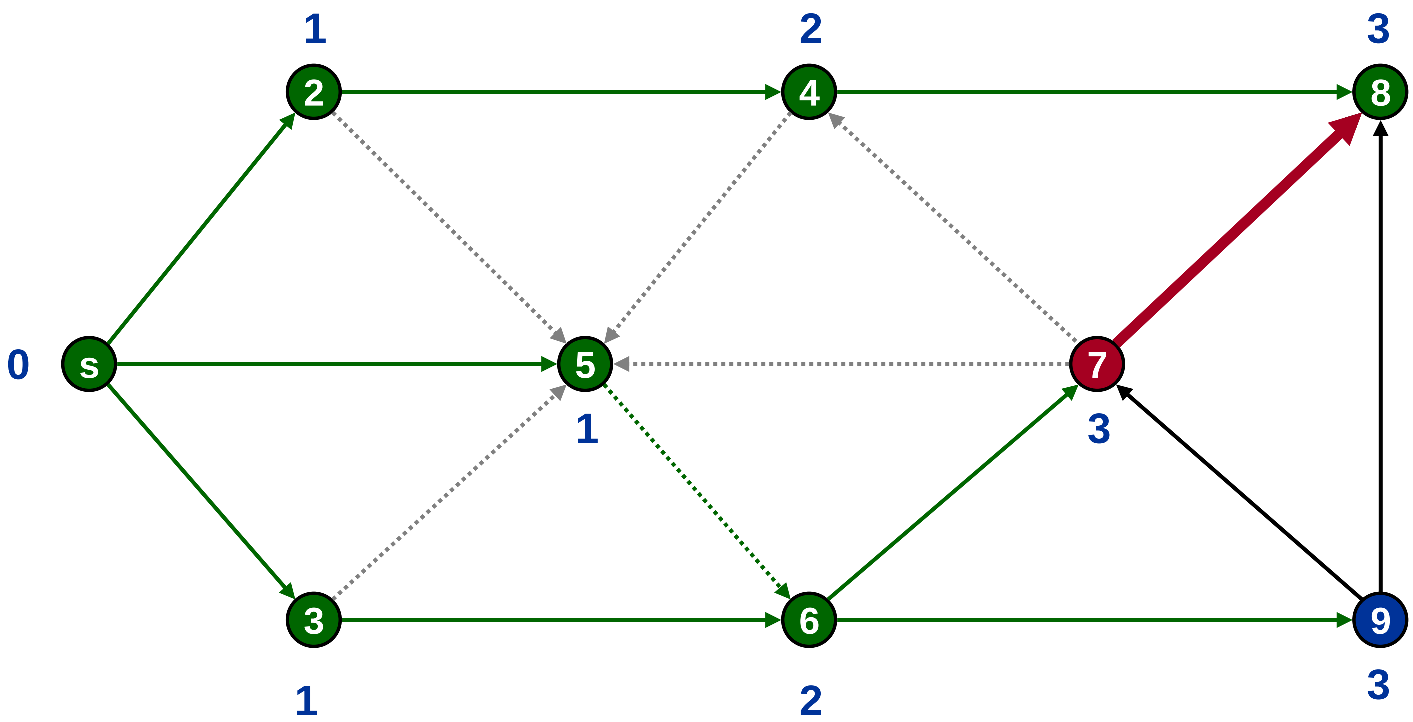
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

Queue: 7 9

Breadth First Search



Undiscovered

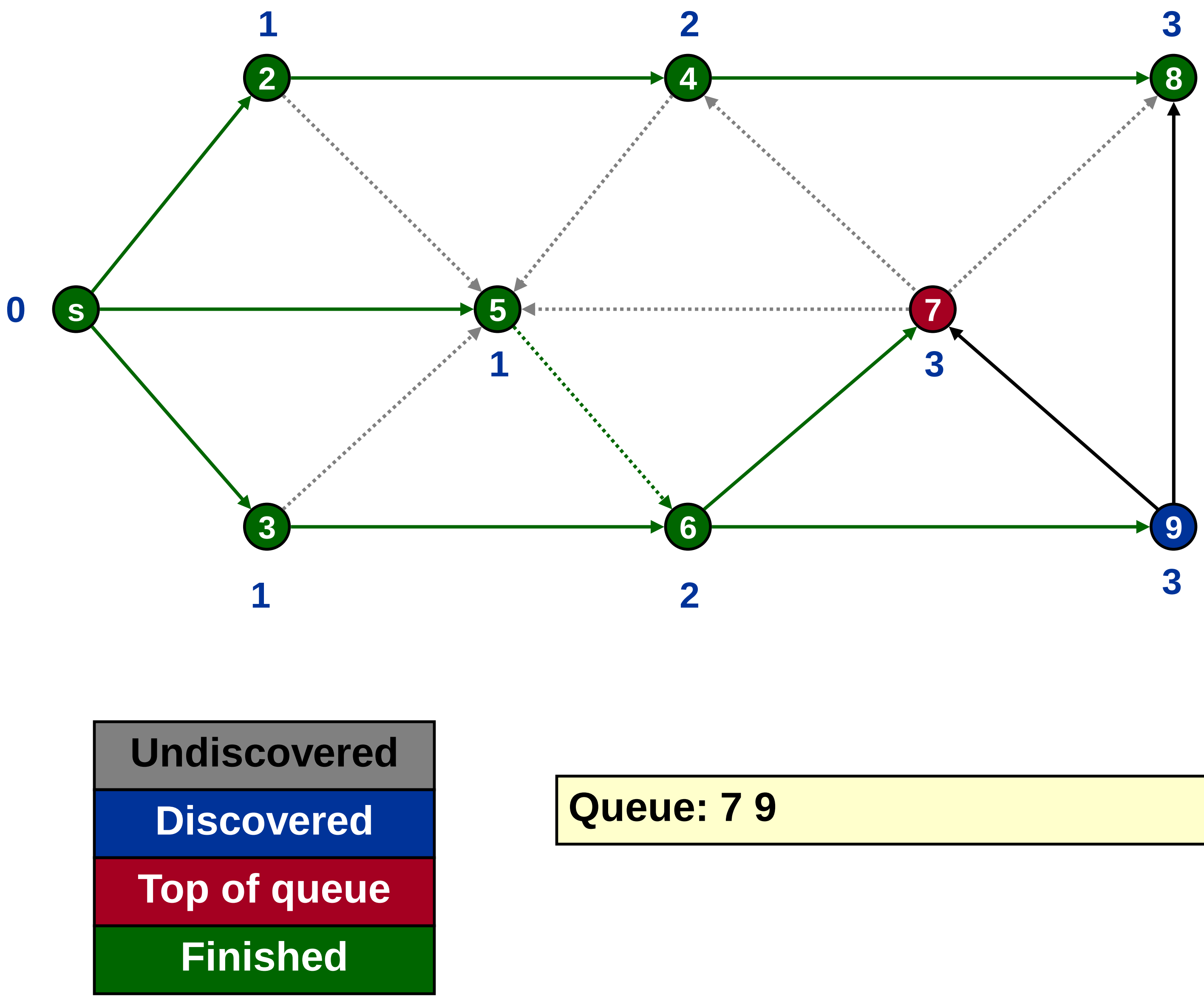
Discovered

Top of queue

Finished

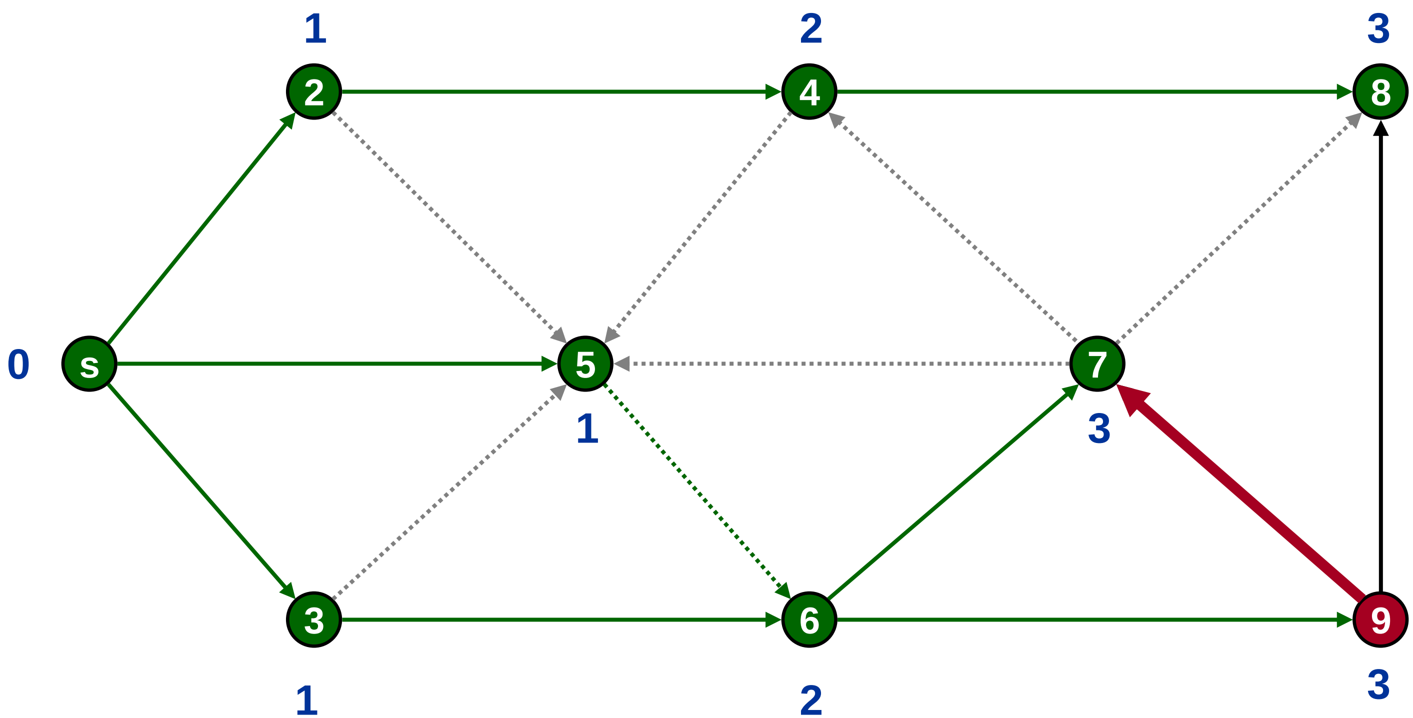
Queue: 7 9

Breadth First Search



Queue: 7 9

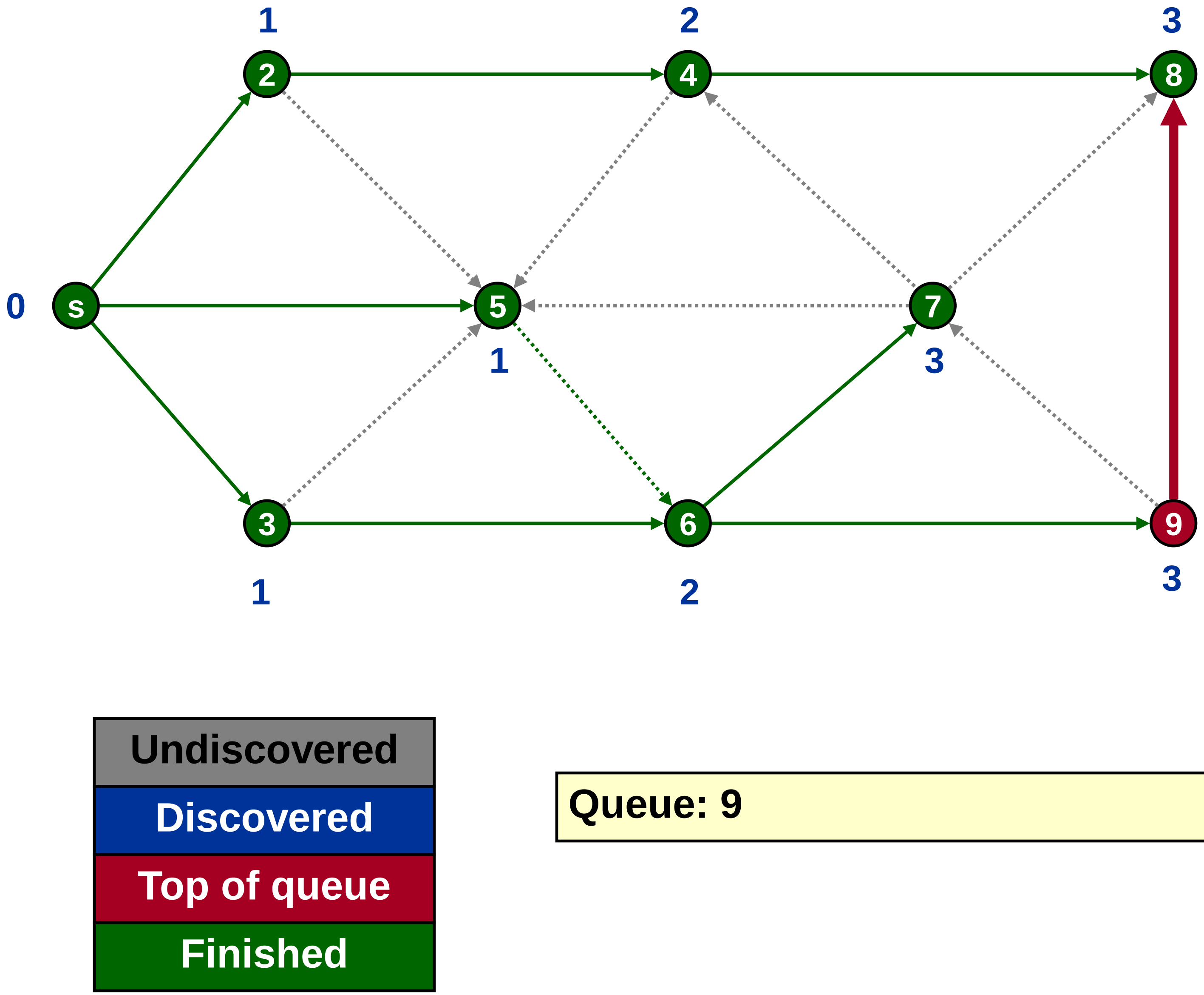
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

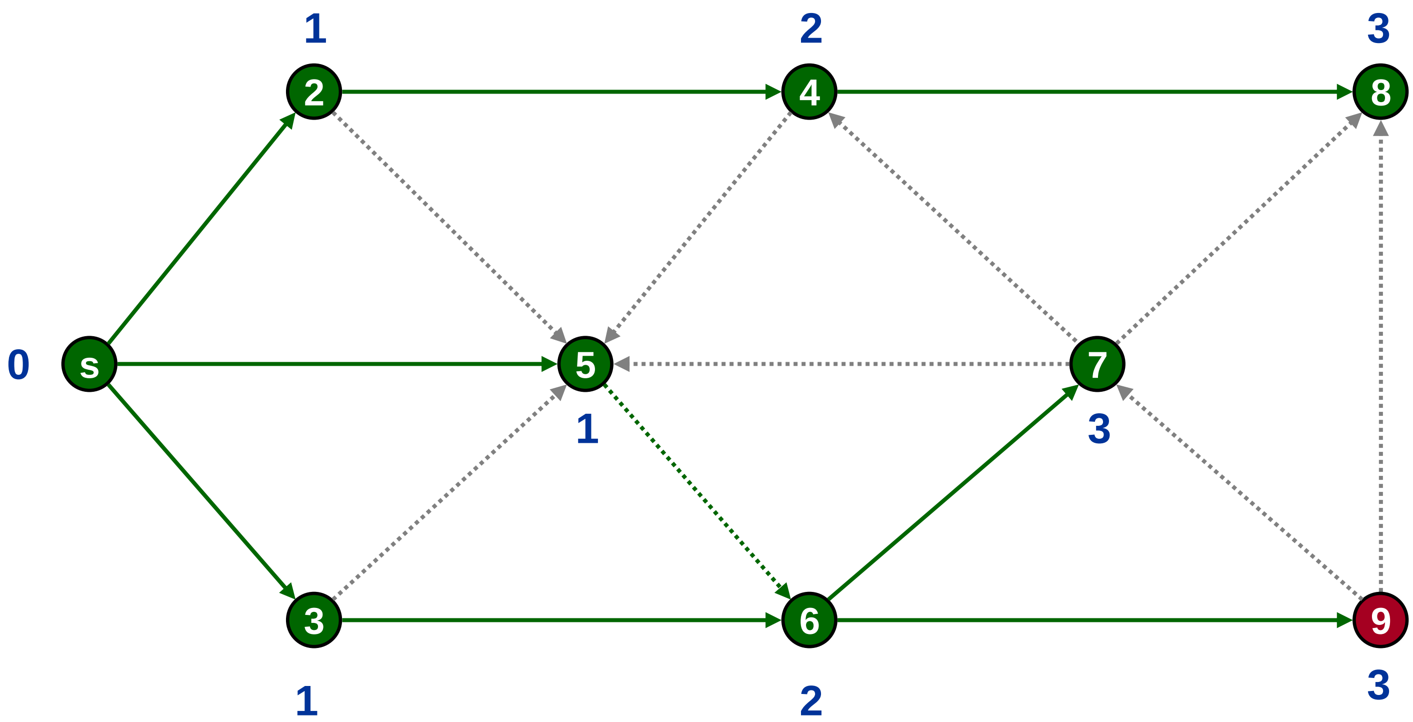
Queue: 9

Breadth First Search



Queue: 9

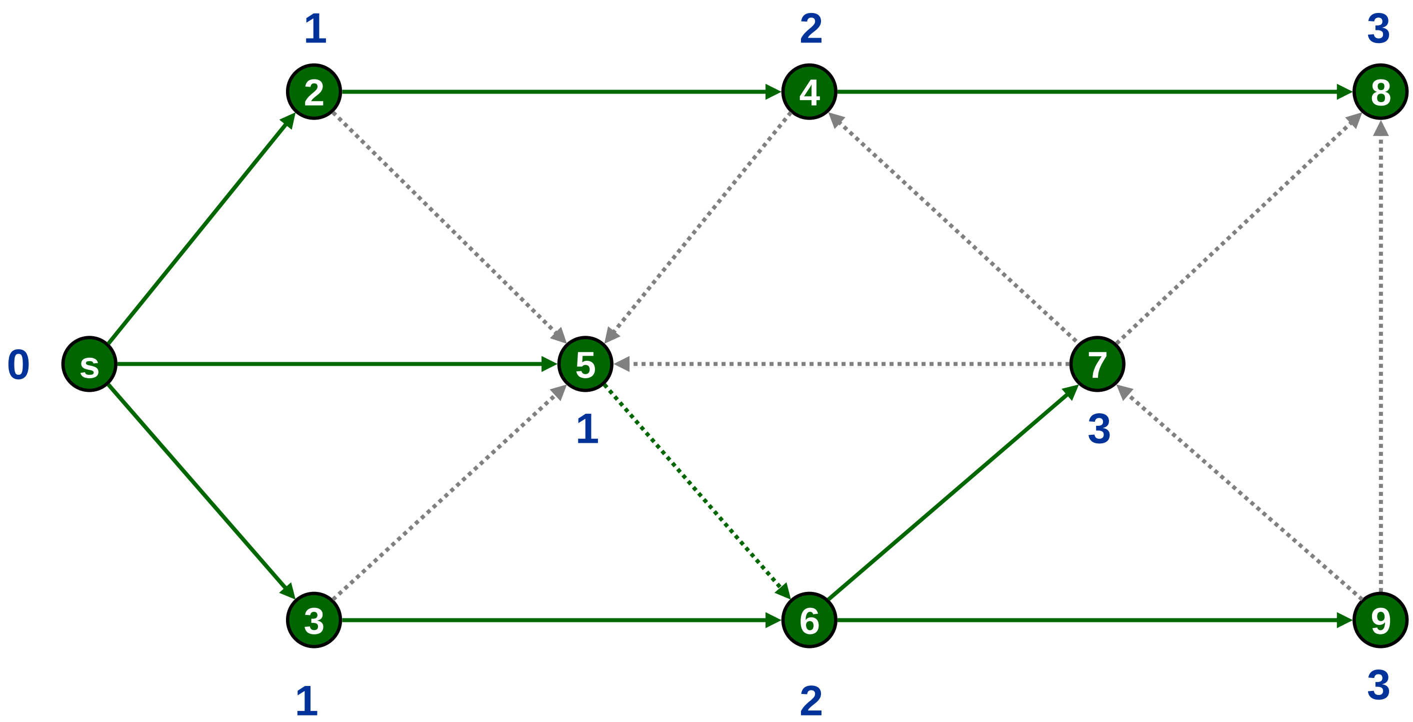
Breadth First Search



Undiscovered
Discovered
Top of queue
Finished

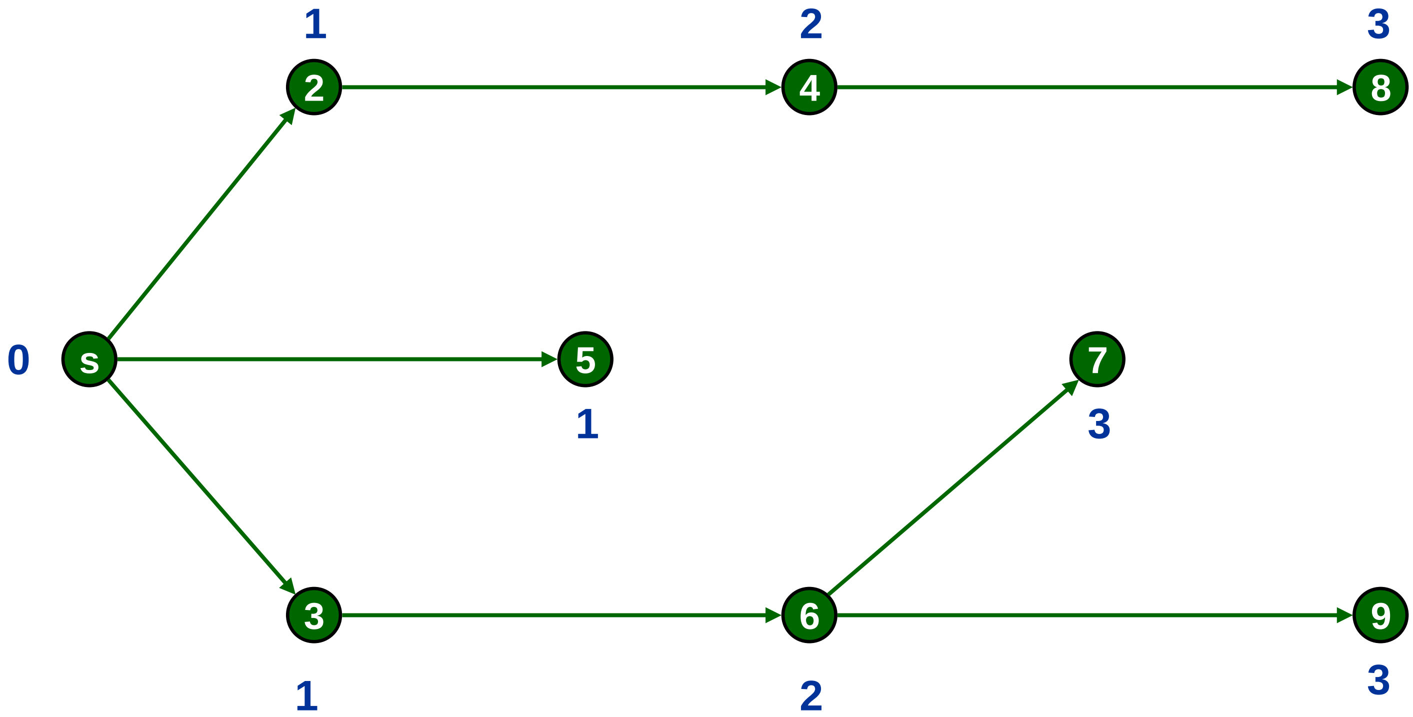
Queue: 9

Breadth First Search



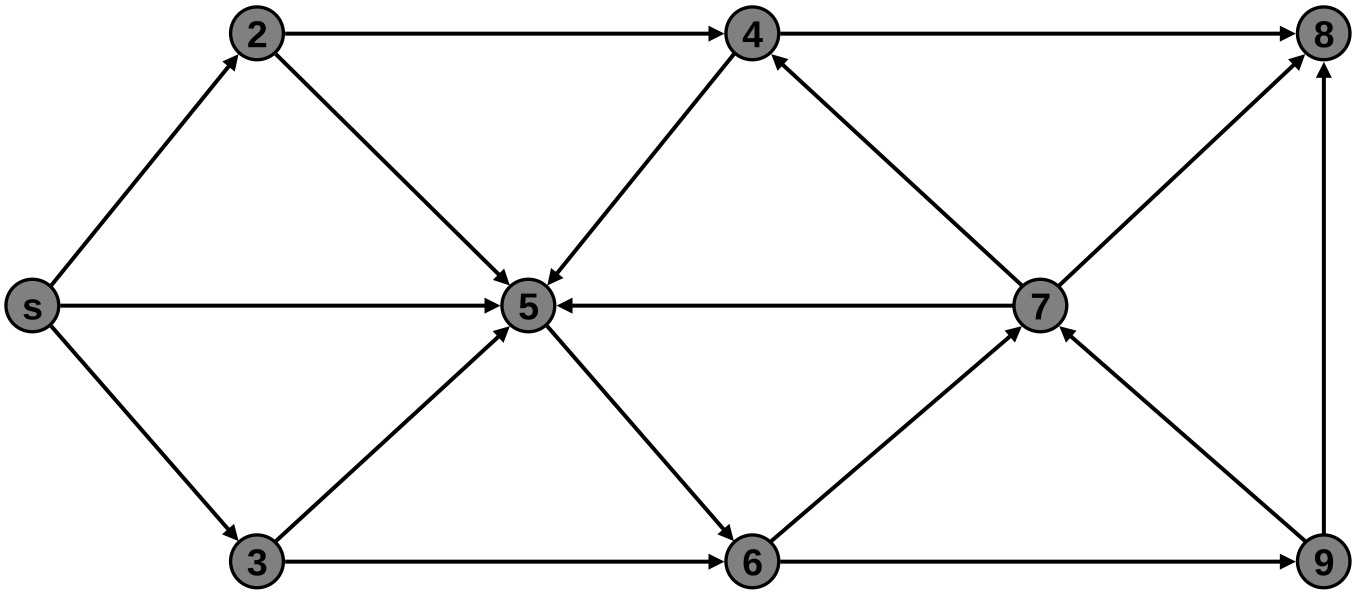
Queue:

Breadth First Search



Level Graph

Depth First Search



- Apply DFS algorithm on the same graph using stack and see what is DFS Tree generated after exhausting whole stack.

BFS/DFS

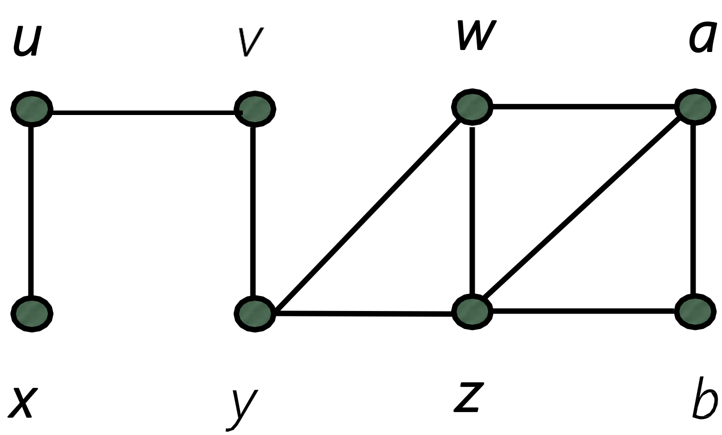
- Time Complexity:
- The operations of enqueuing and dequeuing takes O(1) time, so total time devoted to queue operations is O(V).
- Since sum of the lenghts of all adjacency lists is θ(E) , the total time spent in scanning adjacency lists is O(E).

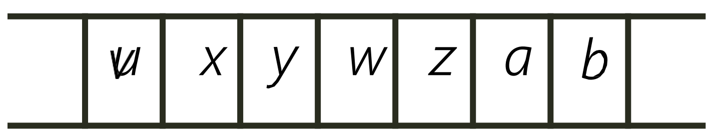
- Total running time of BFS is θ(V+E).
- Same reasoning can be made for DFS.

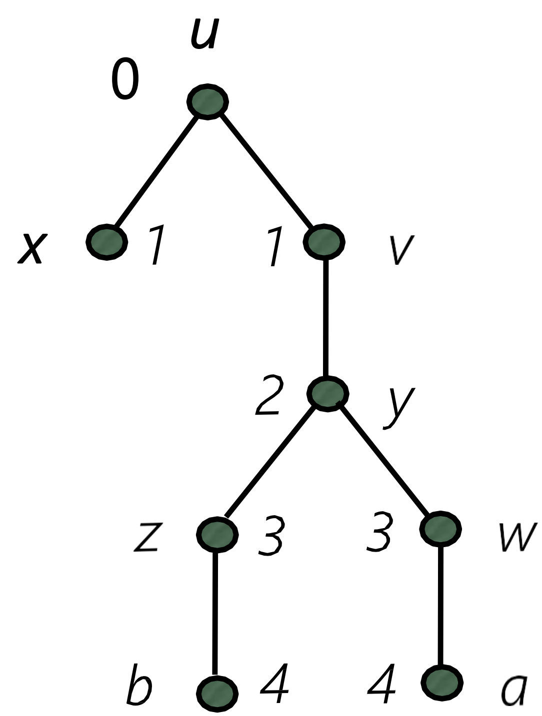
Complexity

<table>
<tr><th></th><th></th><th>Time</th><th>Space</th></tr>
<tr><td rowspan="2">Undirected</td><td>Adj. Matrix</td><td>$O(|V|^2)$</td><td rowspan="2">$O(|V|)$</td></tr>
<tr><td>Adj. List</td><td>$O(|V| + 2|E|)$</td></tr>
</table>

Breadth-first search



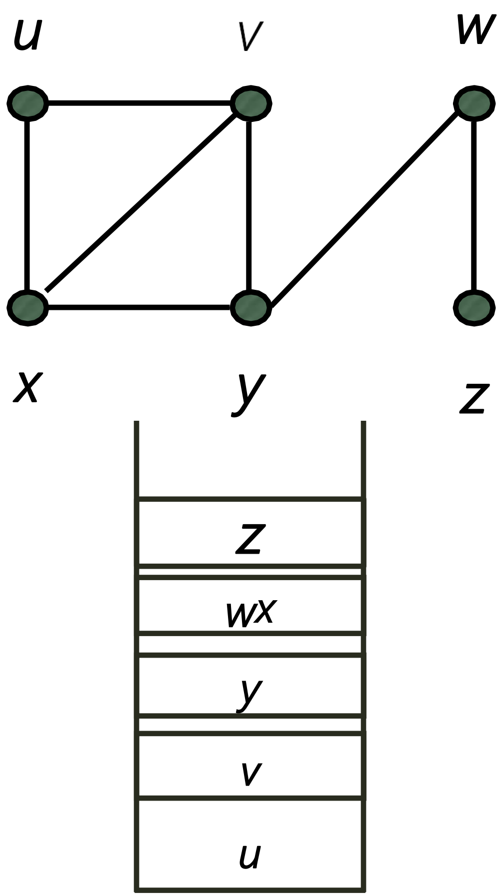


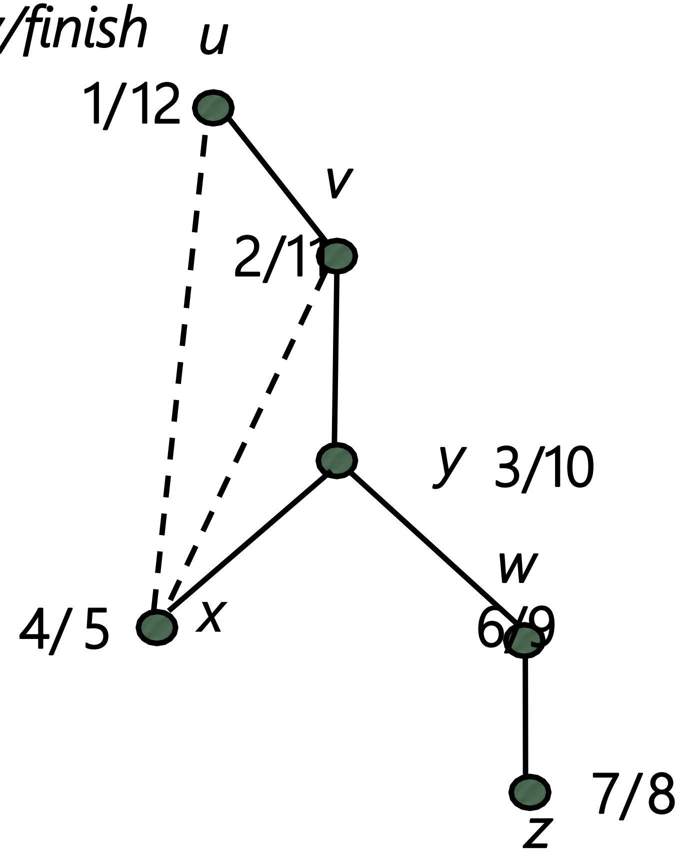


BFS tree

Depth-first search

time = 0 discovery/finish

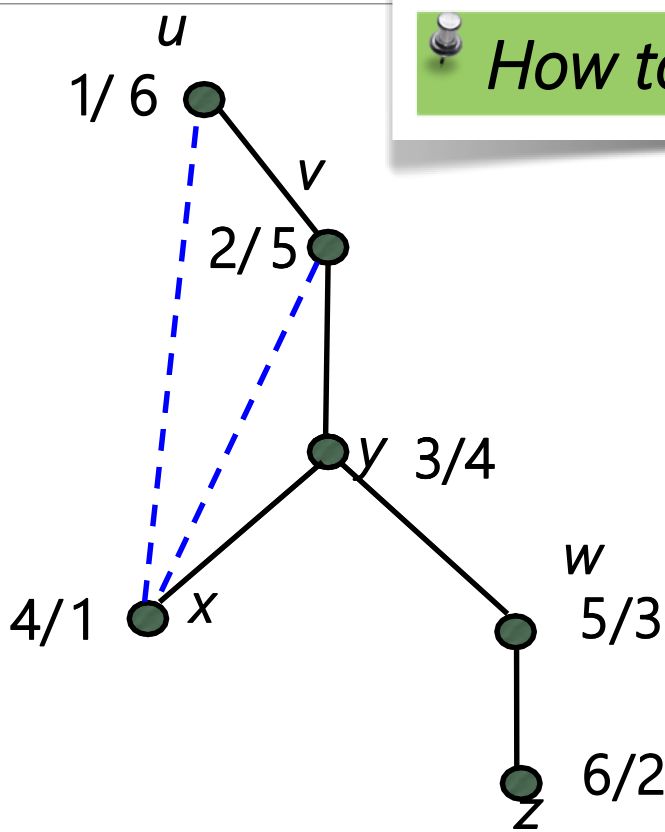




DFS tree

DFS tree: undirected

How to figure out back edges?



DFS tree

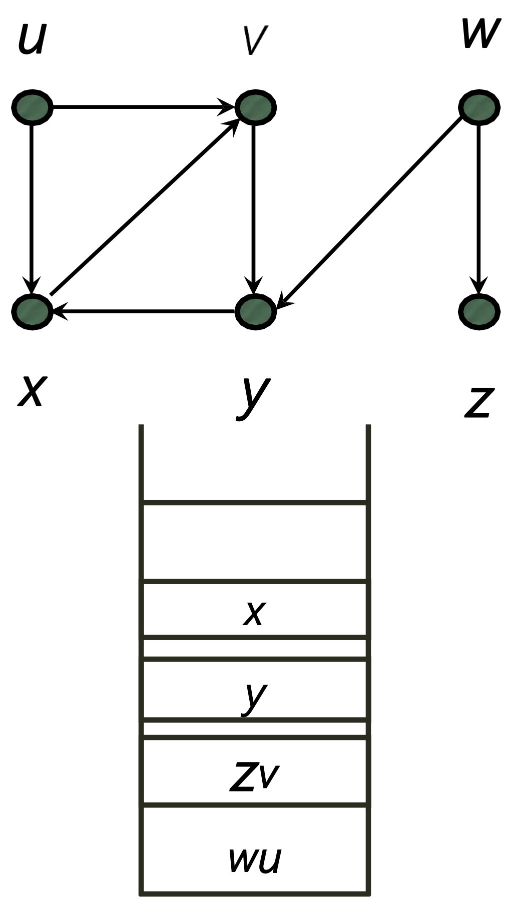
- tree edge:
- back edge:

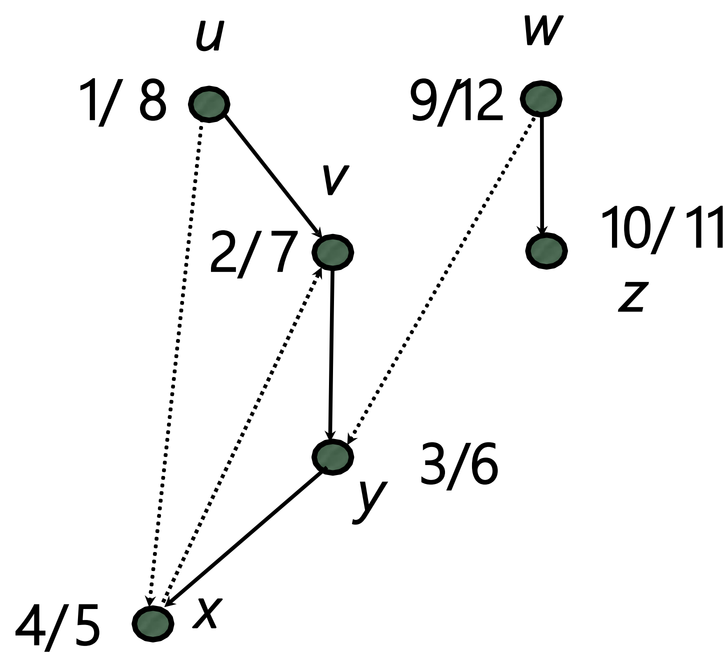
Complexity

		Time	Space
Undirected	Adj. Matrix	$O(\|V\|^2)$	$O(\|V\|)$
	Adj. List	$O(\|V\| + 2\|E\|)$	

DFS tree: directed graph

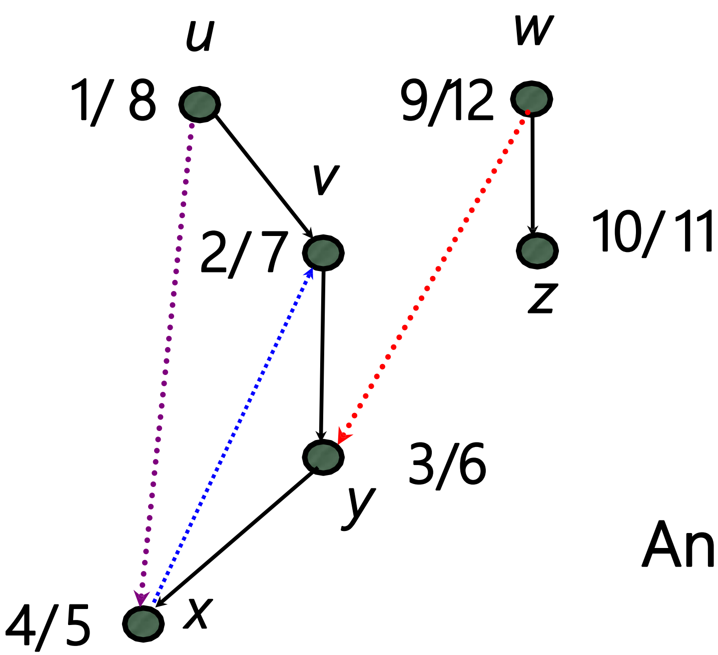
time = 0





DFS trees

DFS tree



- tree edge:
- back edge:
- forward edge:
- cross edge:

An edge (u,v) is a *back edge* iff
finish(u) < finish(v)

Graph Acyclicity

NO back edges!

Connectivity

Definition
An undirected graph is **connected**, if there is a path between any pair of vertices.
A **connected component** is a subgraph that is internally connected but has no edges to the remaining vertices.

#trees == #connected components