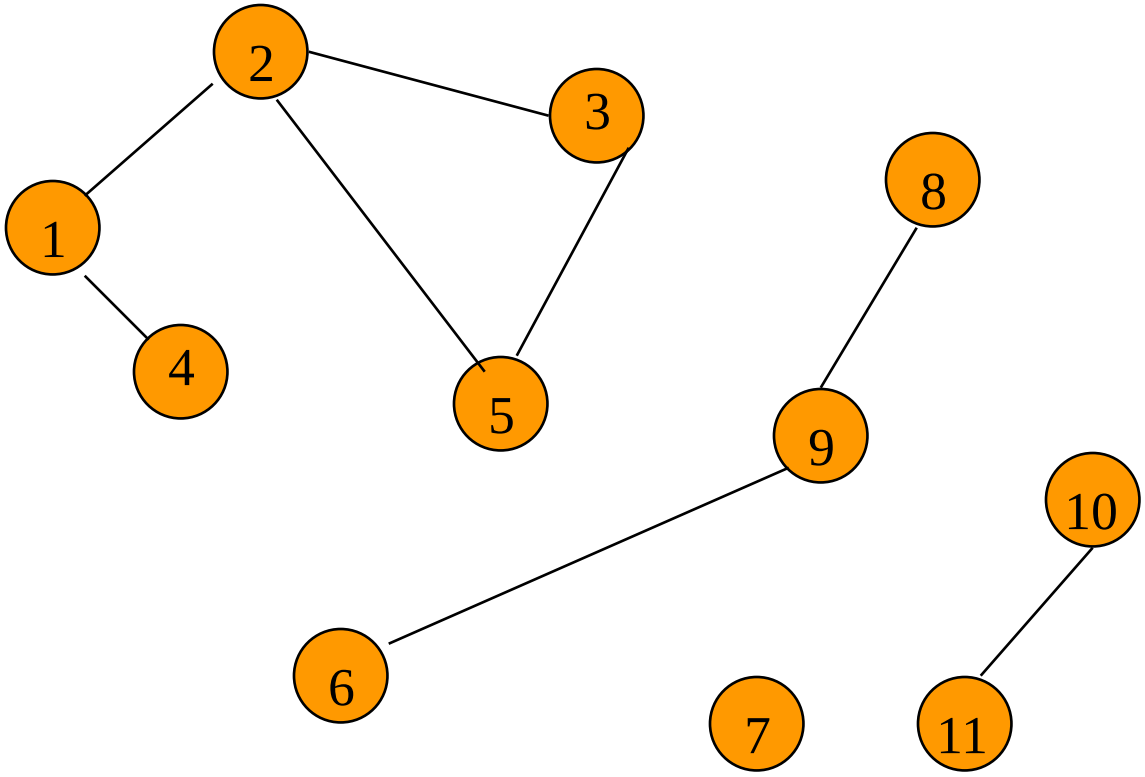
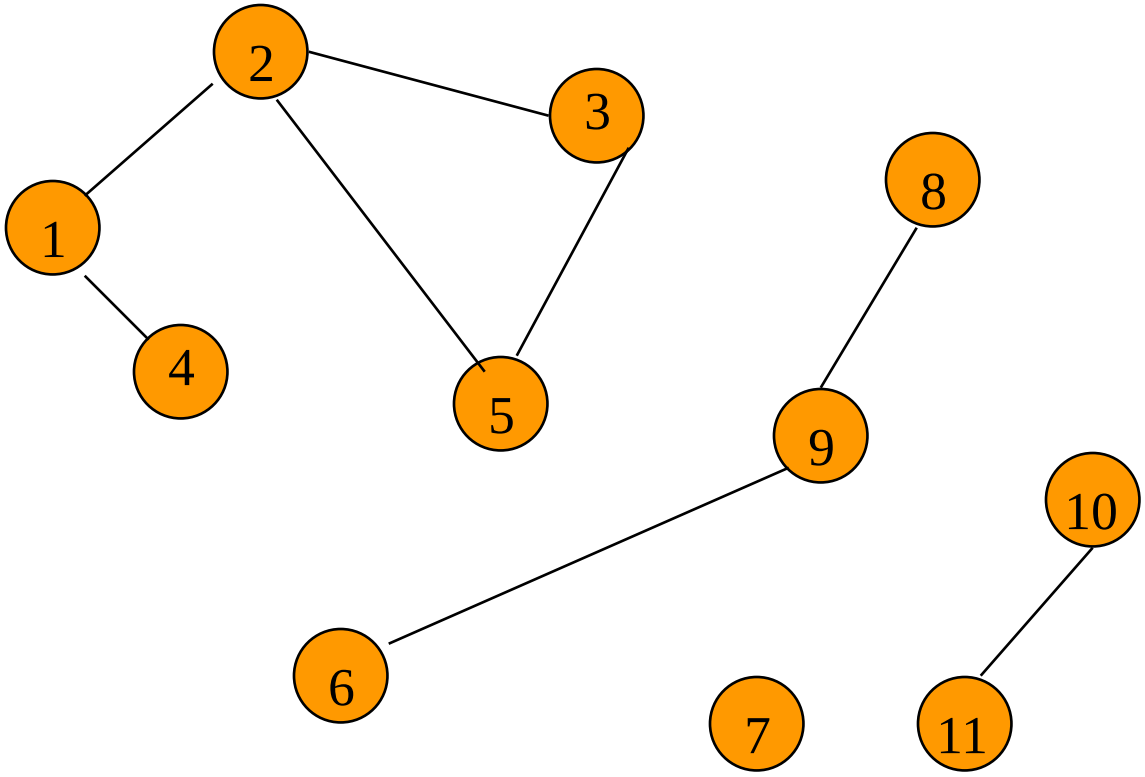
Graph Search Methods

- A vertex u is reachable from vertex v iff there is a path from v to u.



Graph Search Methods

- A search method starts at a given vertex v and visits/labels/marks every vertex that is reachable from v.



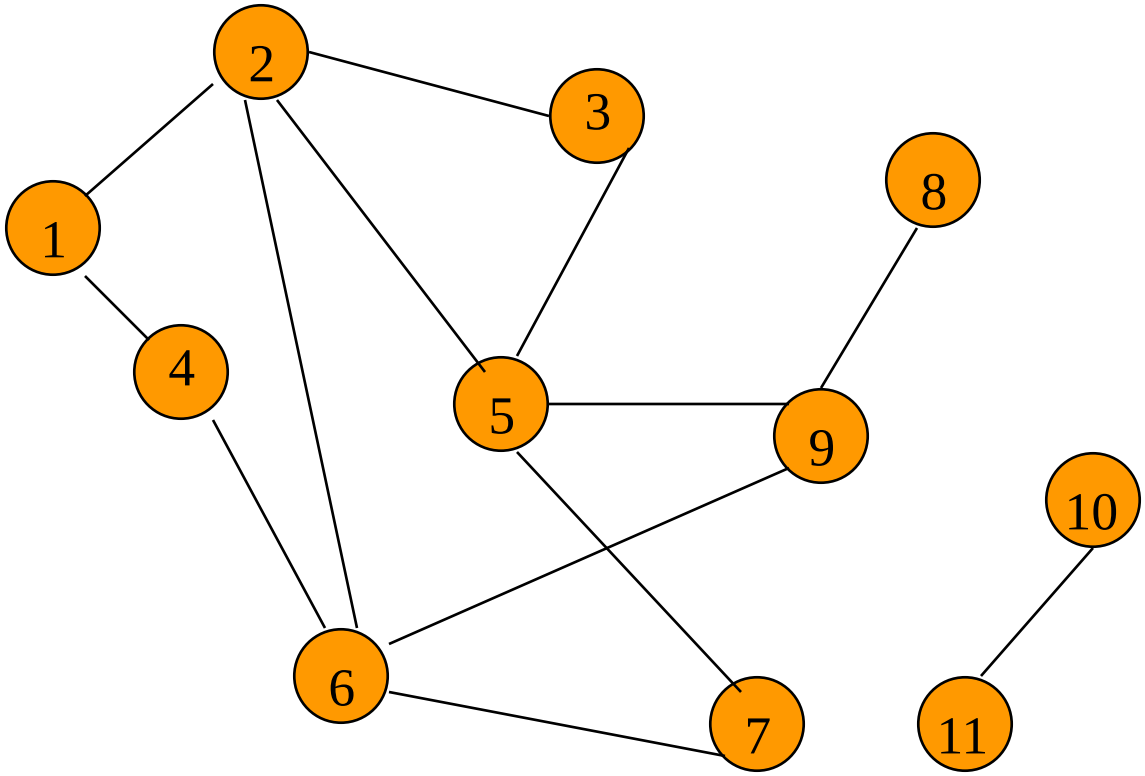
Graph Search Methods

- Many graph problems solved using a search method.
  - Path from one vertex to another.
  - Is the graph connected?
  - Find a spanning tree.
  - Etc.
- Commonly used search methods:
  - Breadth-first search.
  - Depth-first search.

Breadth-First Search

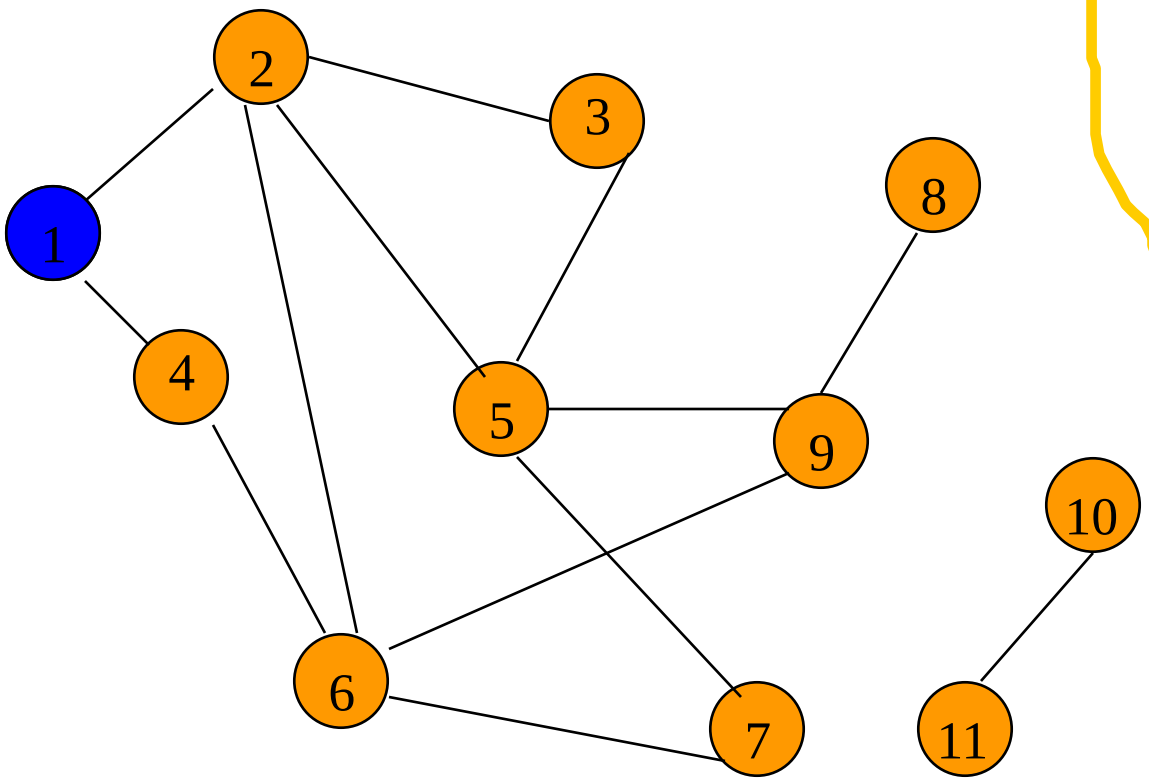
- Visit start vertex and put it into a FIFO queue.
- Repeatedly remove a vertex from the queue, visit its unvisited adjacent vertices, put newly visited vertices into the queue.

Breadth-First Search Example



Start search at vertex 1.

Breadth-First Search Example

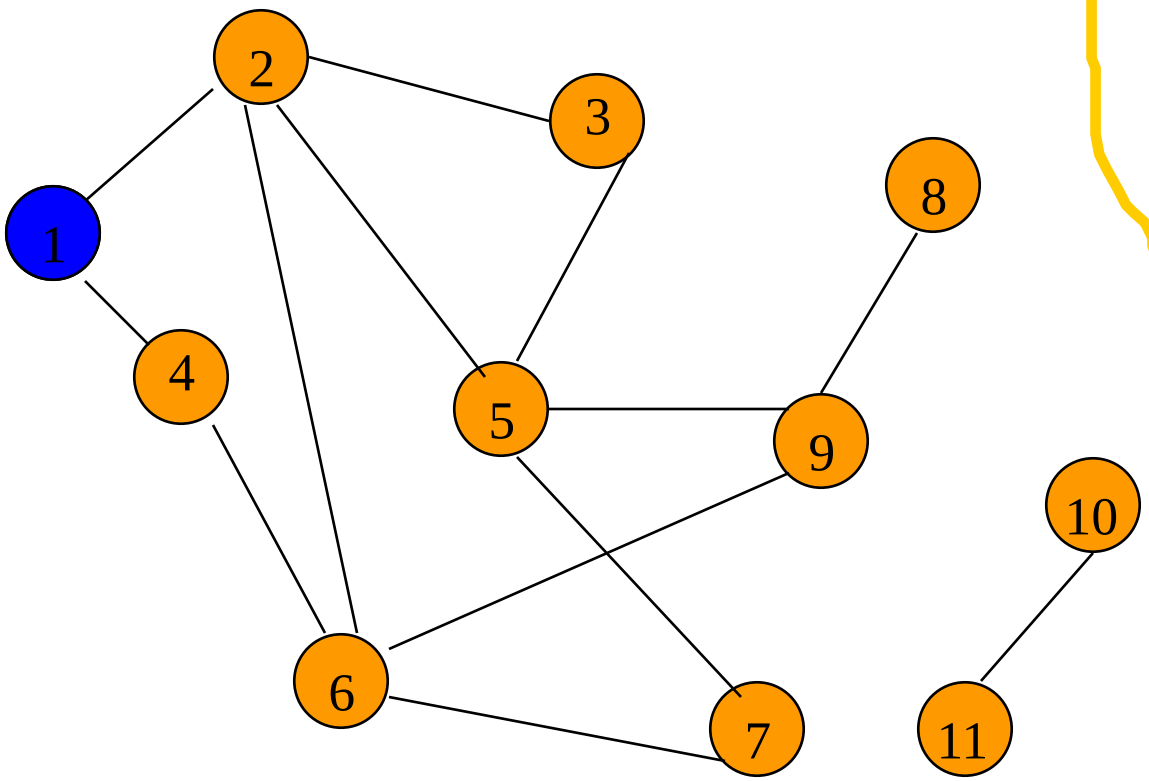


FIFO Queue

1

Visit/mark/label start vertex and put in a FIFO queue.

Breadth-First Search Example

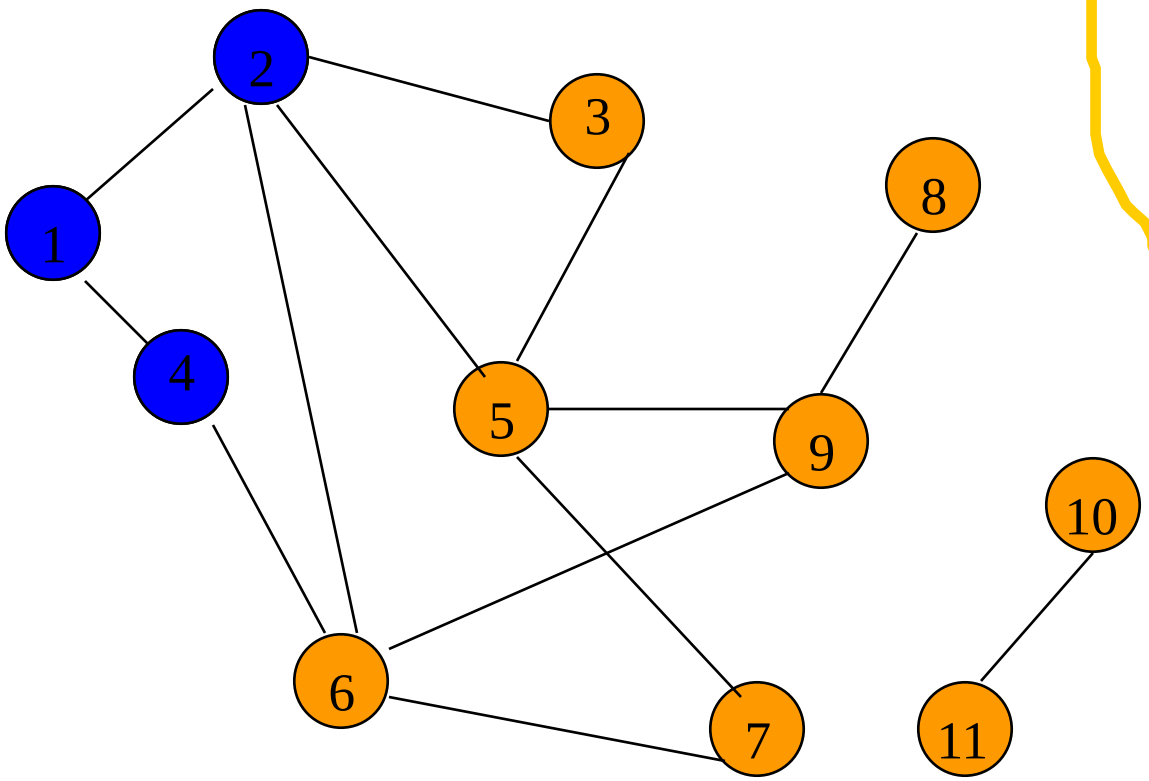


FIFO Queue

1

Remove 1 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

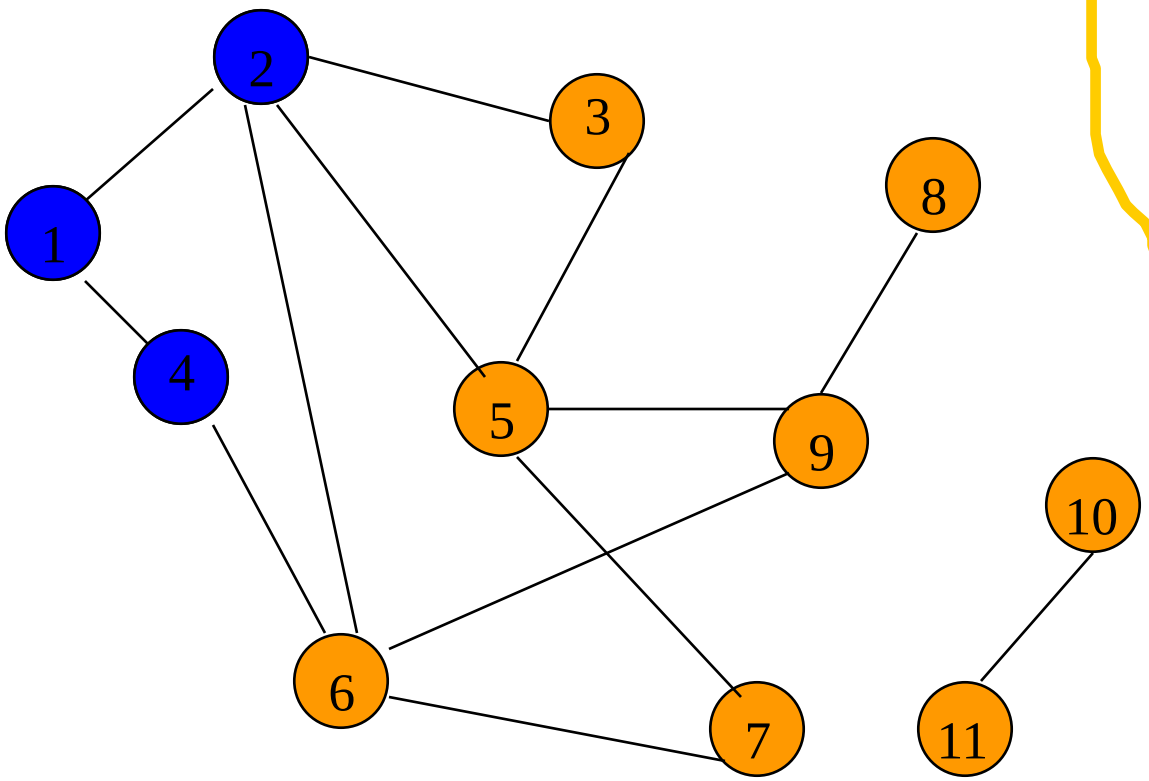


FIFO Queue

2 4

Remove 1 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

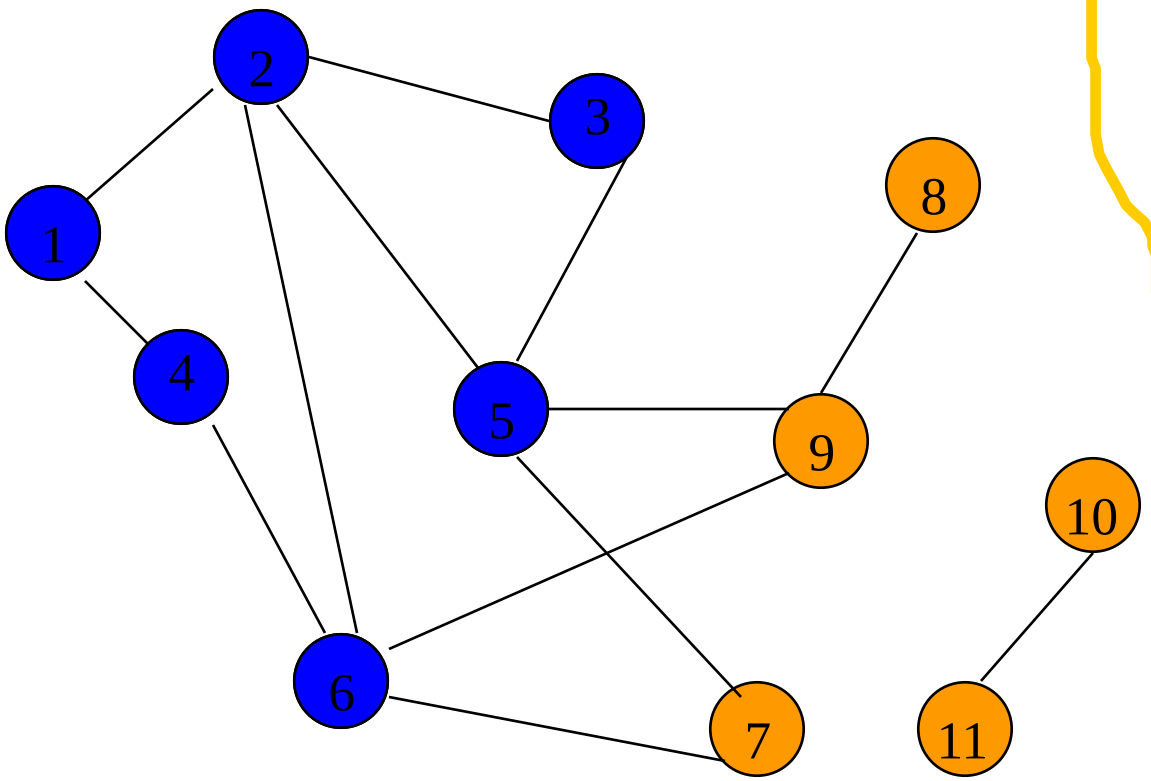


FIFO Queue

2 4

Remove 2 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

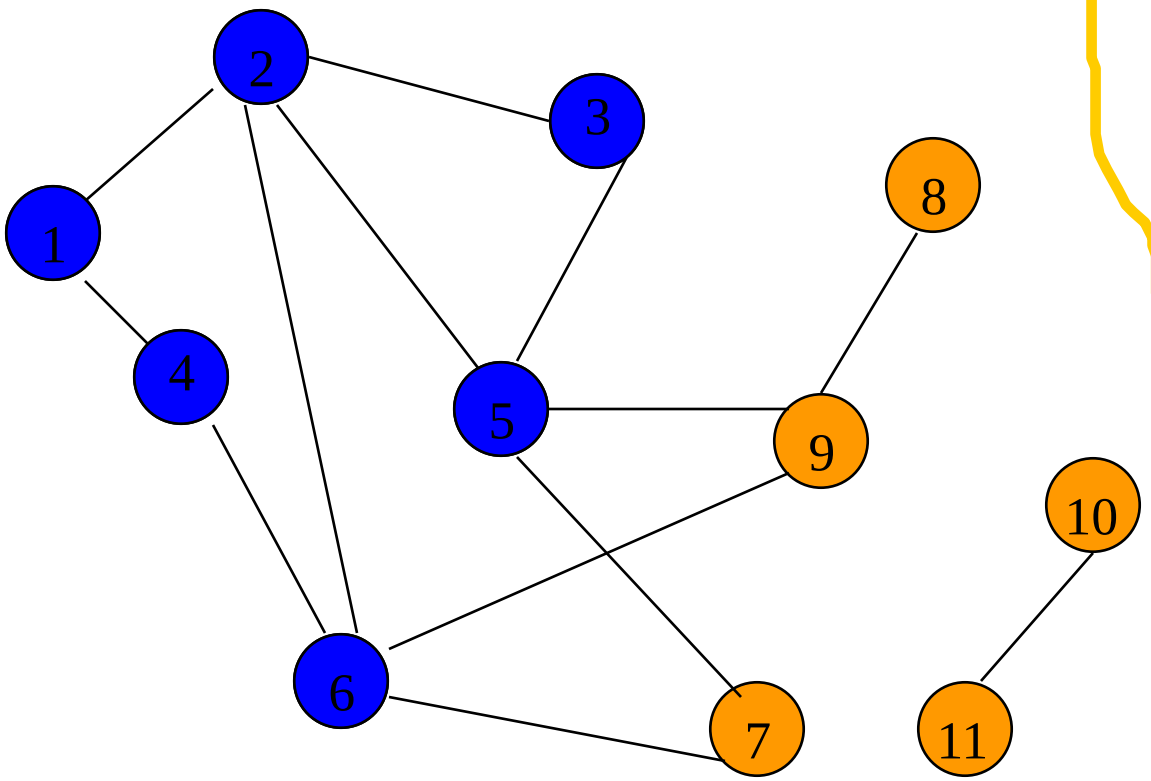


FIFO Queue

4 5 3 6

Remove 2 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

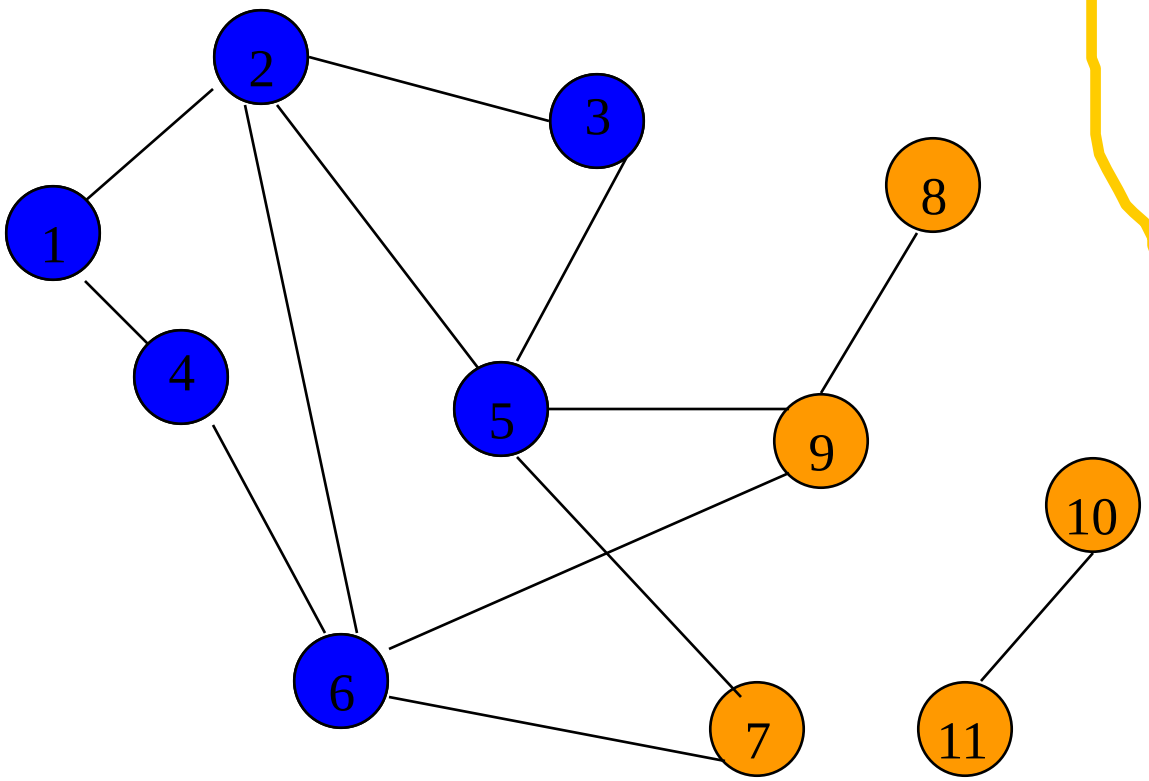


FIFO Queue

4 5 3 6

Remove 4 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

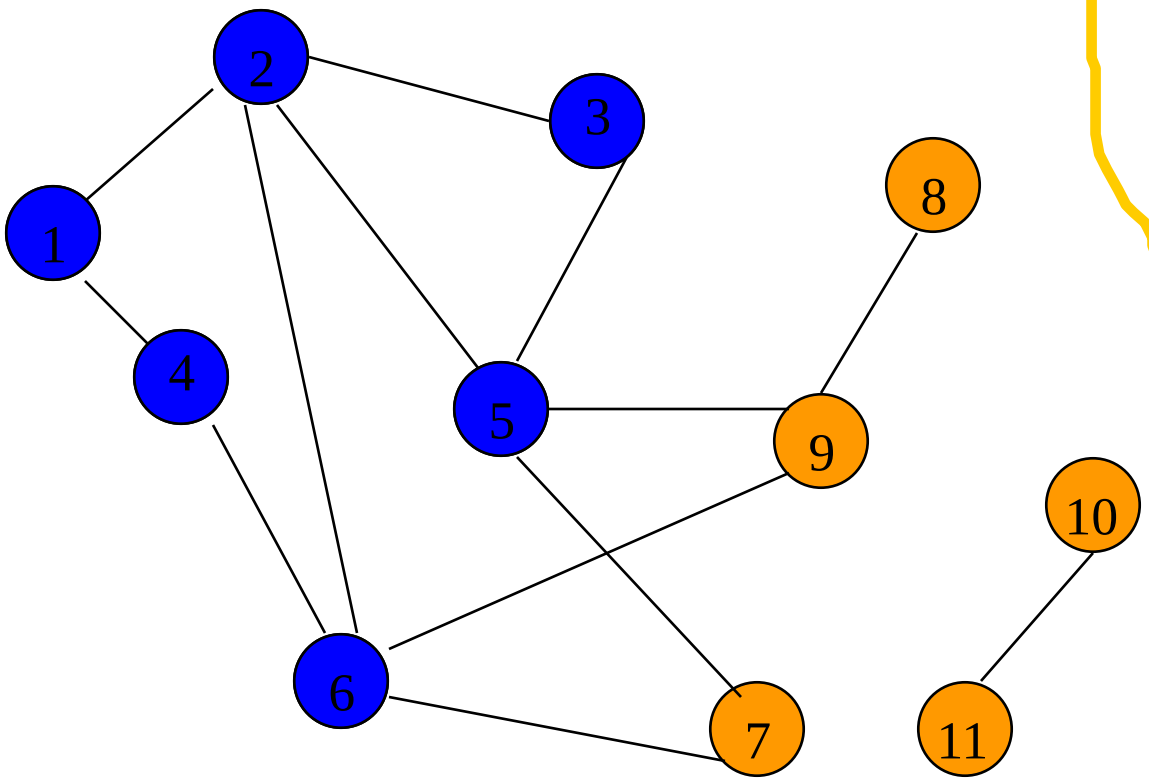


FIFO Queue

5 3 6

Remove 4 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

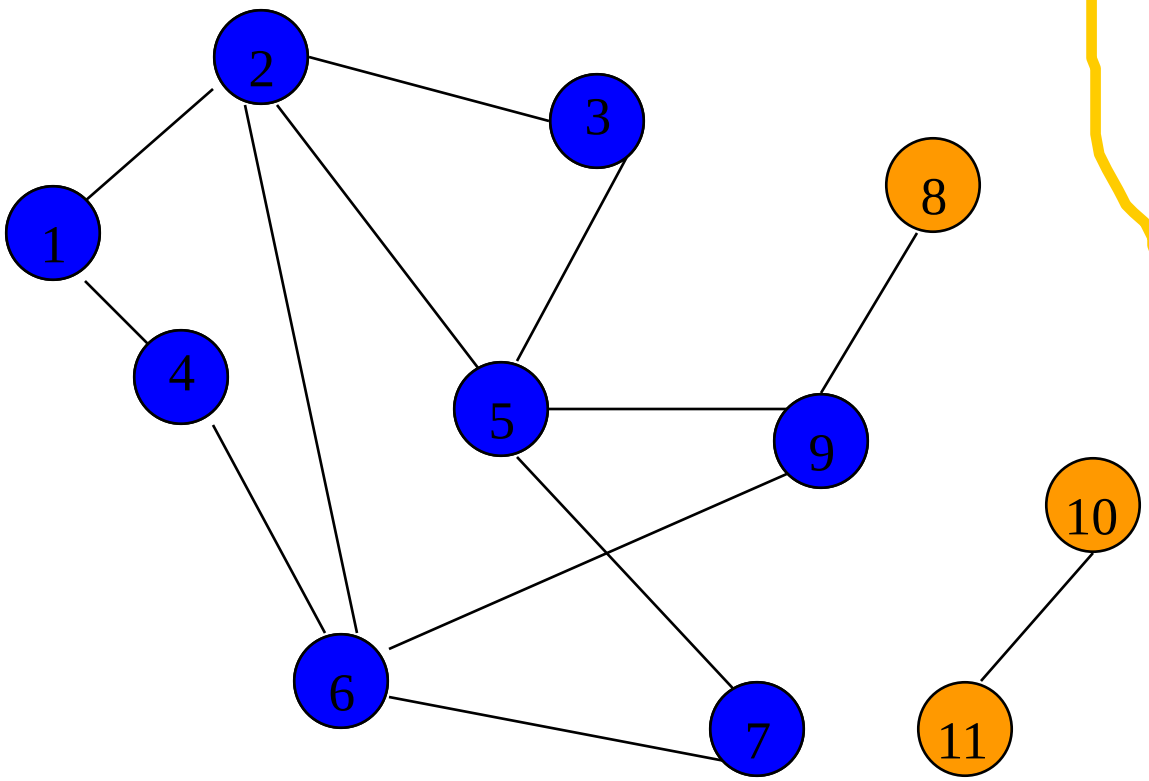


FIFO Queue

5 3 6

Remove 5 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

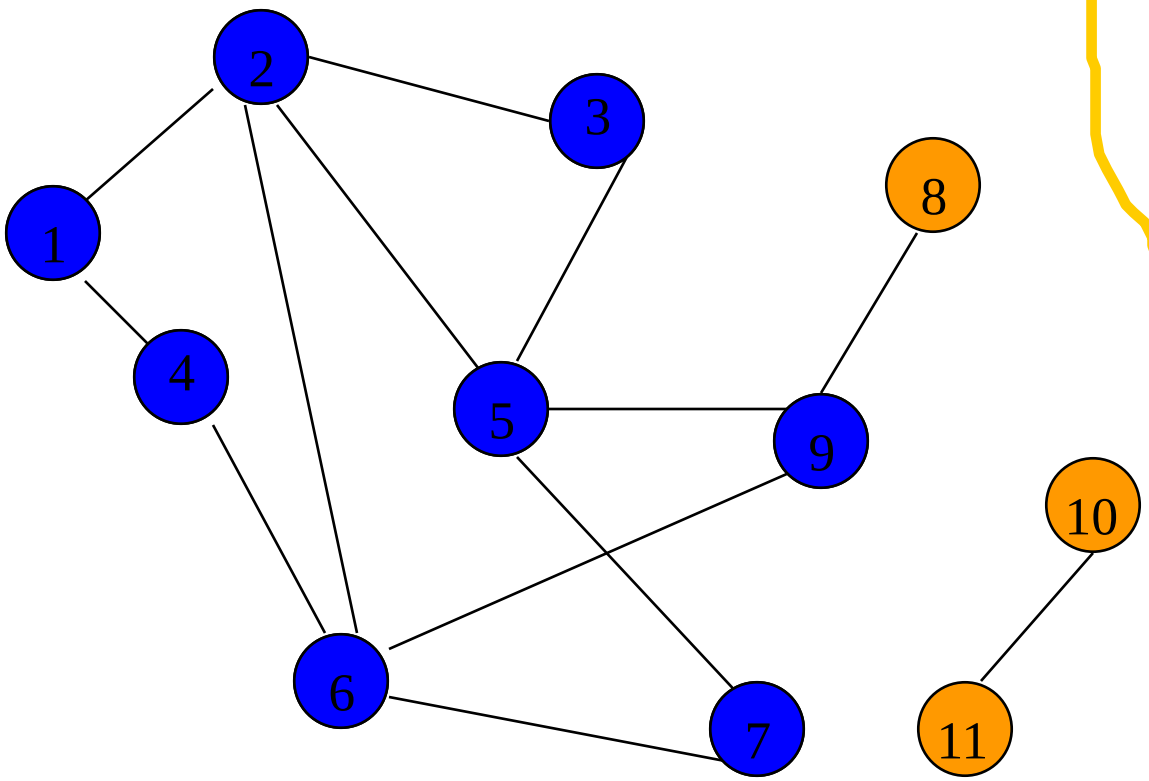


FIFO Queue

3 6 9 7

Remove 5 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

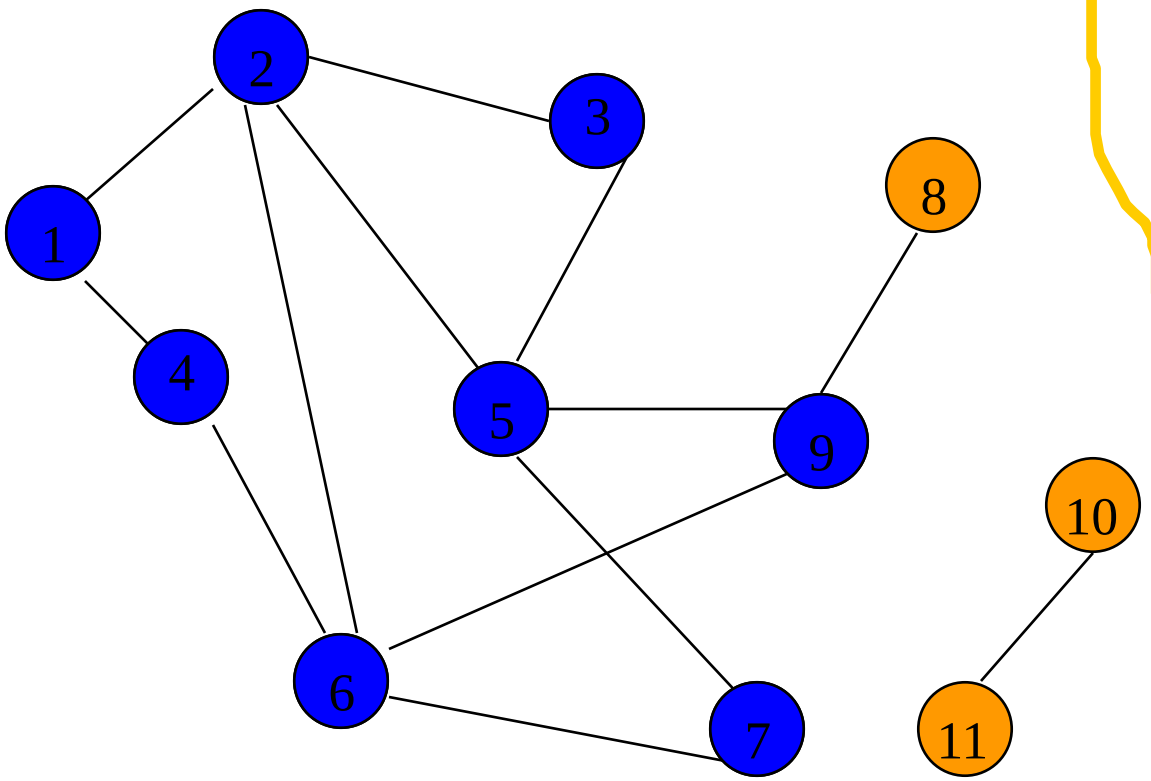


FIFO Queue

3 6 9 7

Remove 3 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

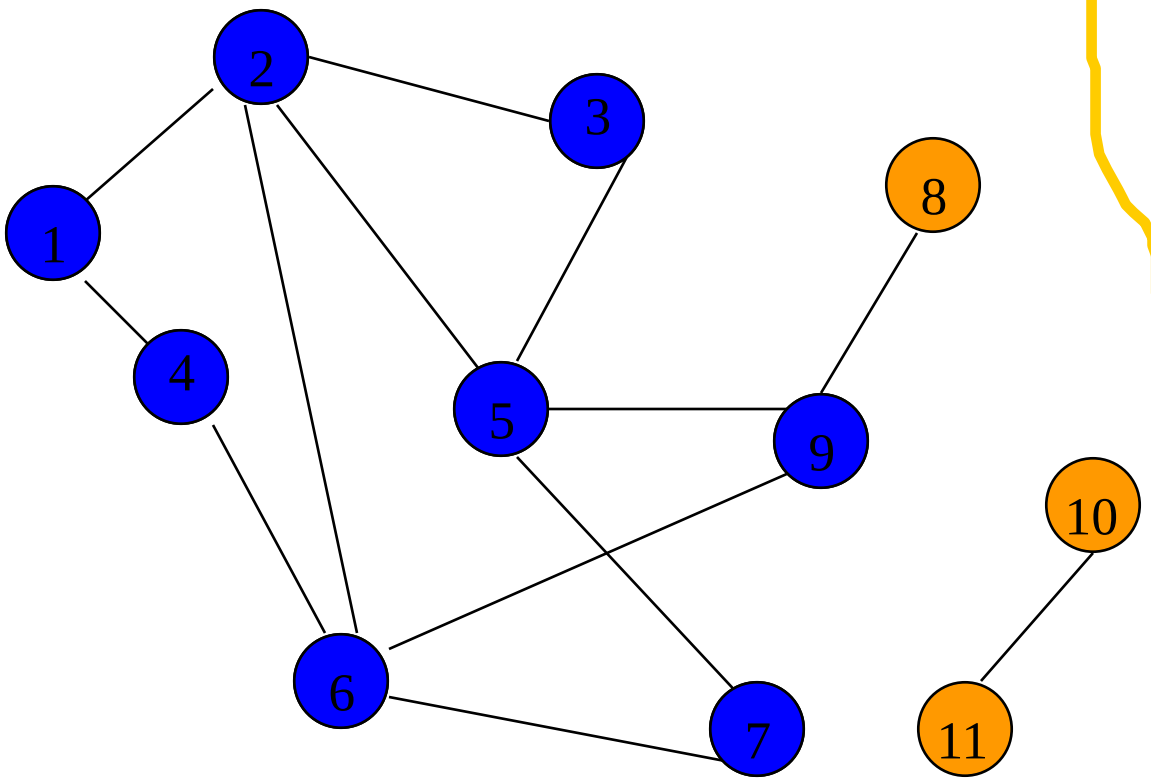


FIFO Queue

6 9 7

Remove 3 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

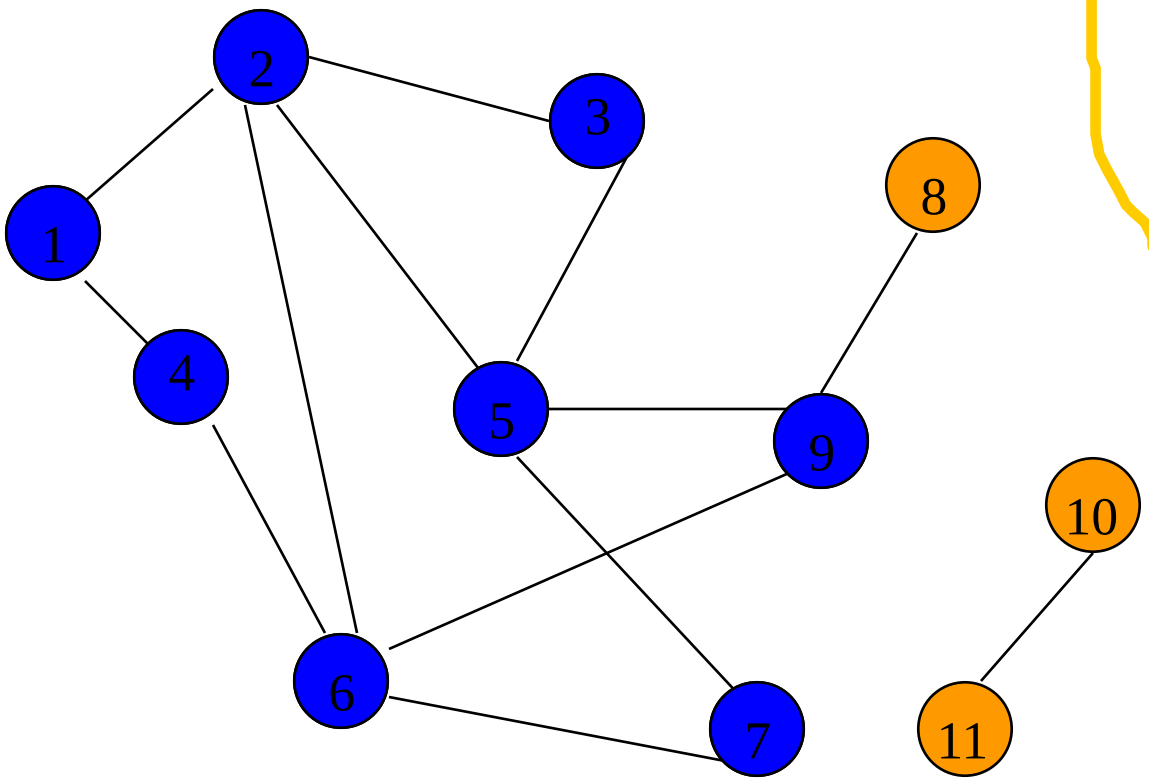


FIFO Queue

6 9 7

Remove 6 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example



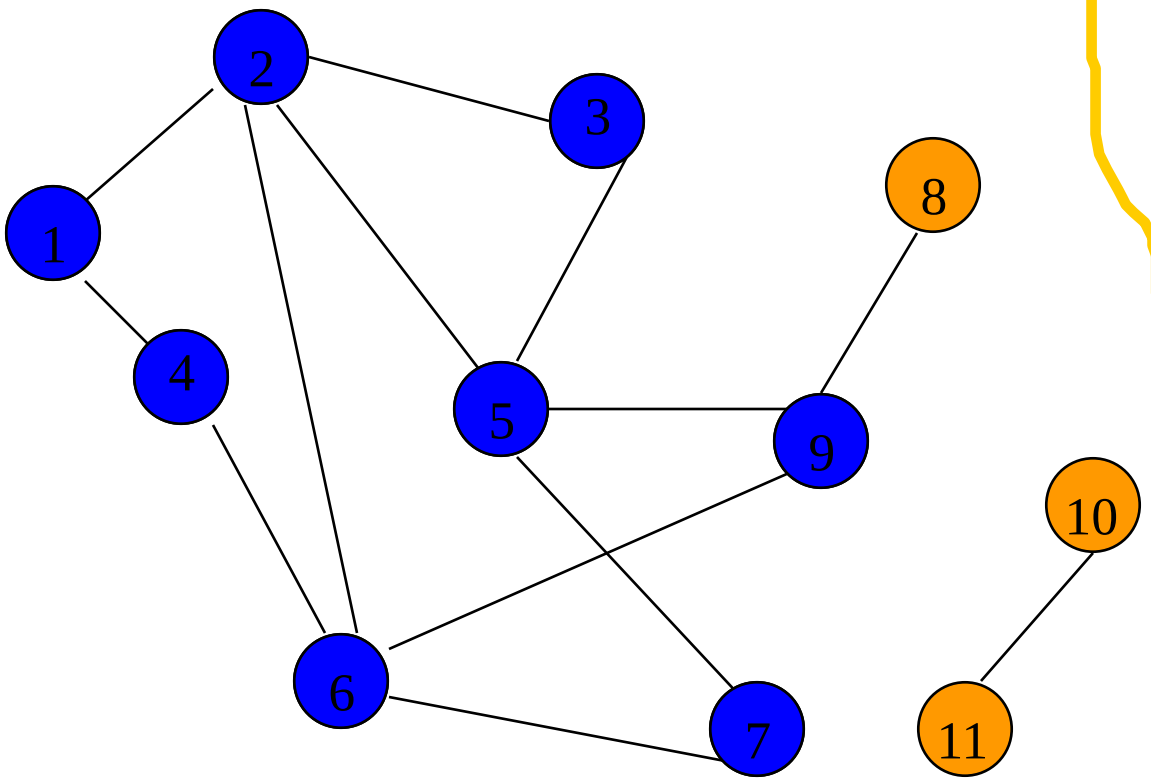
FIFO Queue

9 7

Remove 6 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example



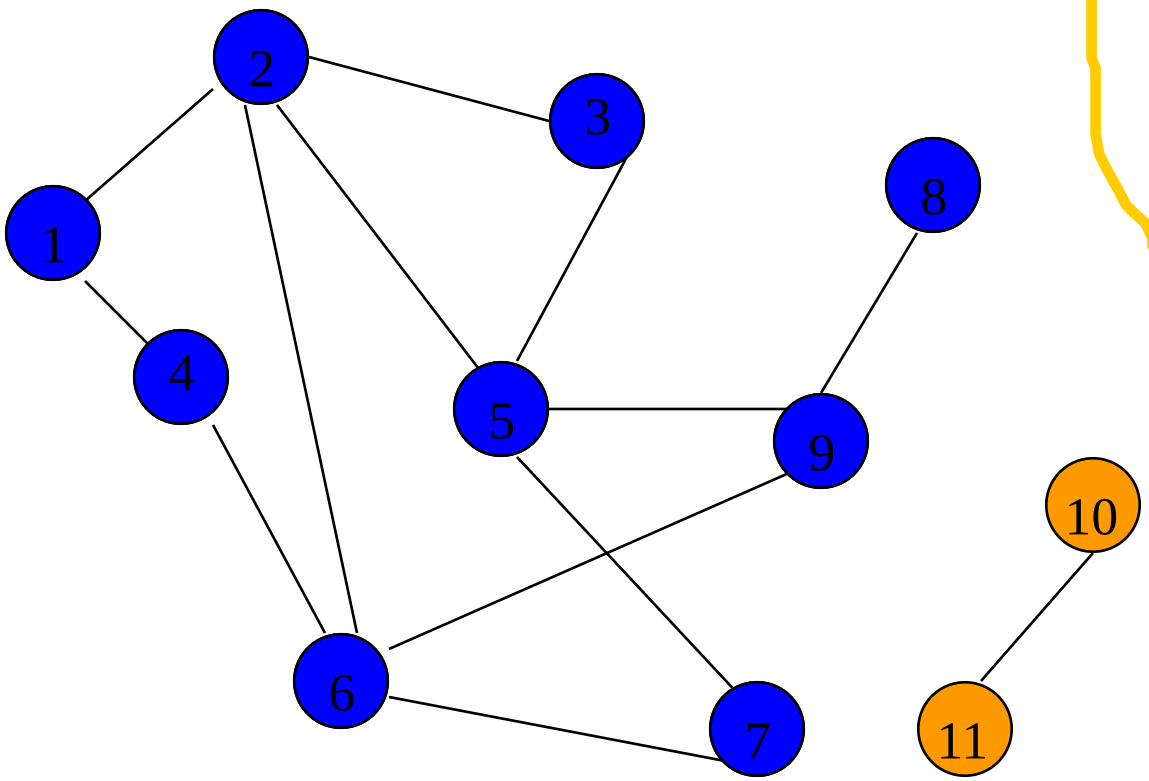
FIFO Queue

9 7

Remove 9 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example



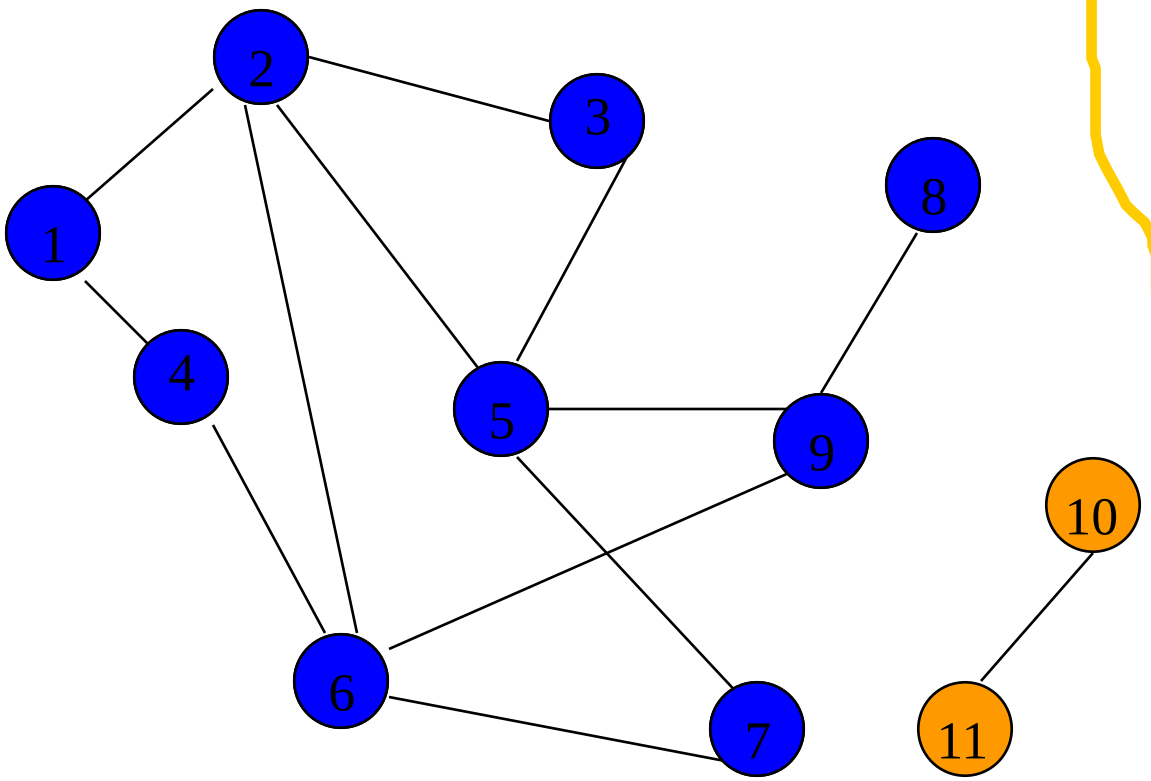
FIFO Queue

7 8

Remove 9 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example



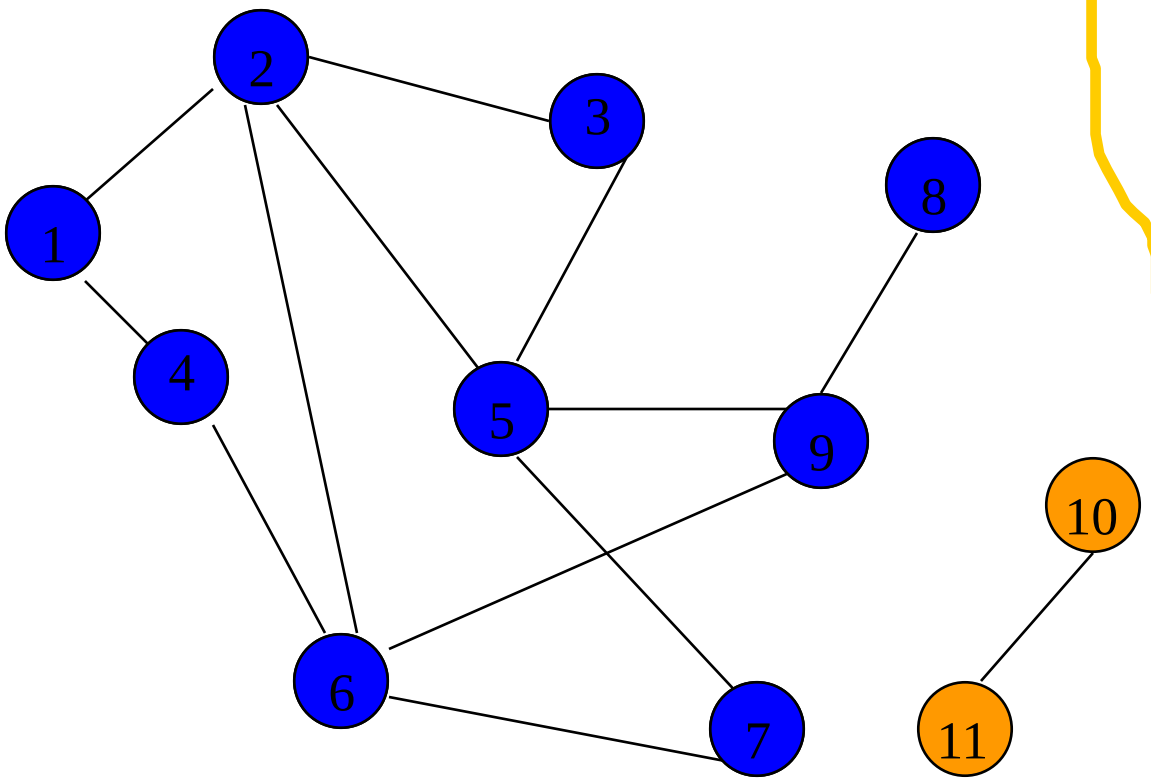
FIFO Queue

7 8

Remove 7 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example



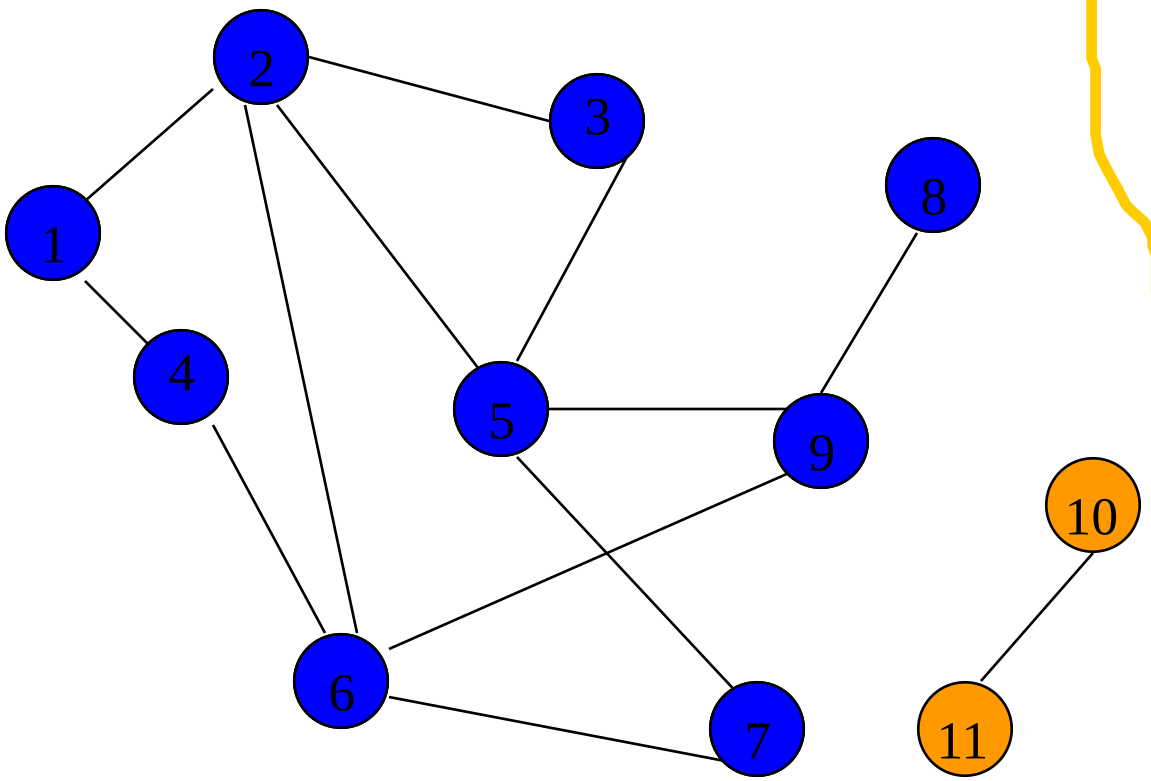
FIFO Queue

8

Remove 7 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example



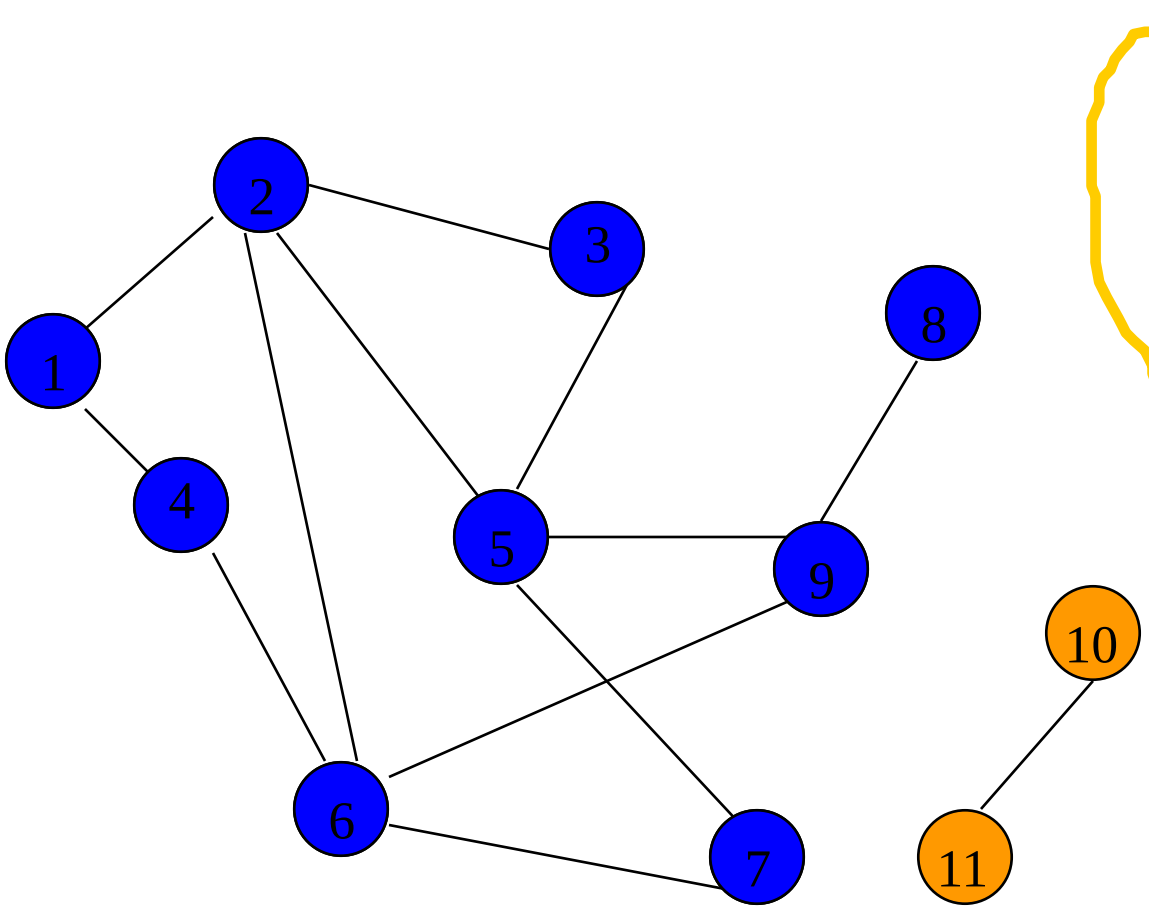
FIFO Queue

8

Remove 8 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example



FIFO Queue

Queue is empty. Search terminates.

Breadth-First Search Property

- All vertices reachable from the start vertex (including the start vertex) are visited.

Time Complexity

- Each visited vertex is put on (and so removed from) the queue exactly once.
- When a vertex is removed from the queue, we examine its adjacent vertices.
  - O(n) if adjacency matrix used [search in 1 row]
  - O(vertex degree) if adjacency lists used
- Total time
  - O(mn), where m is number of vertices in the component that is searched (adjacency matrix)

Time Complexity

- O(n + sum of component vertex degrees) (adj. lists)
  = O(n + number of edges in component)

Path From Vertex v To Vertex u

- Start a breadth-first search at vertex v.
- Terminate when vertex u is visited or when Q becomes empty (whichever occurs first).
- Time
  - $O(n^2)$ when adjacency matrix used
  - $O(n+e)$ when adjacency lists used (e is number of edges)

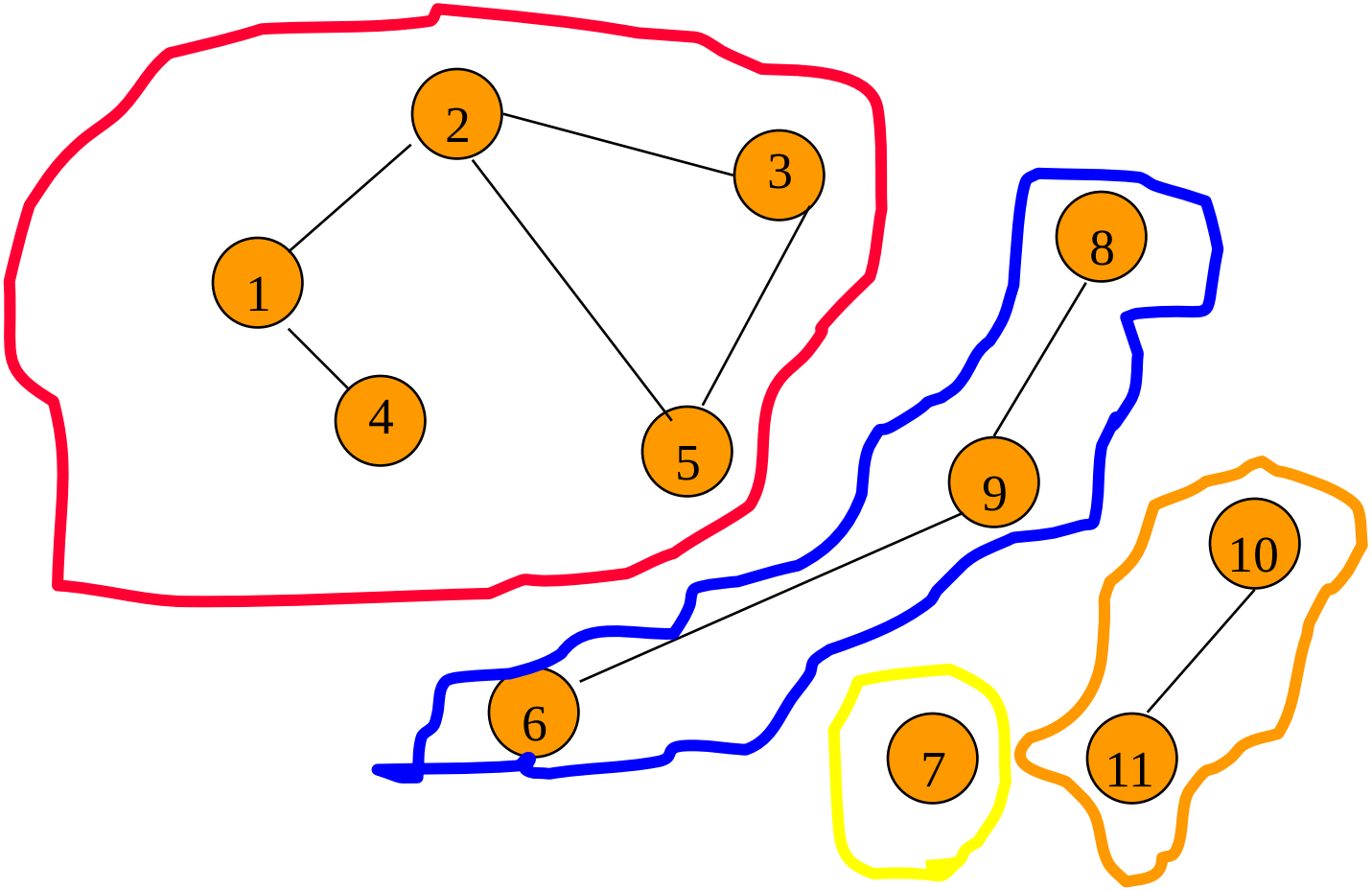
Is The Graph Connected?

- Start a breadth-first search at any vertex of the graph.
- Graph is connected iff all n vertices get visited.
- Time
  - $O(n^2)$ when adjacency matrix used
  - $O(n+e)$ when adjacency lists used (e is number of edges)

Connected Components

- Start a breadth-first search at any as yet unvisited vertex of the graph.
- Newly visited vertices (plus edges between them) define a component.
- Repeat until all vertices are visited.

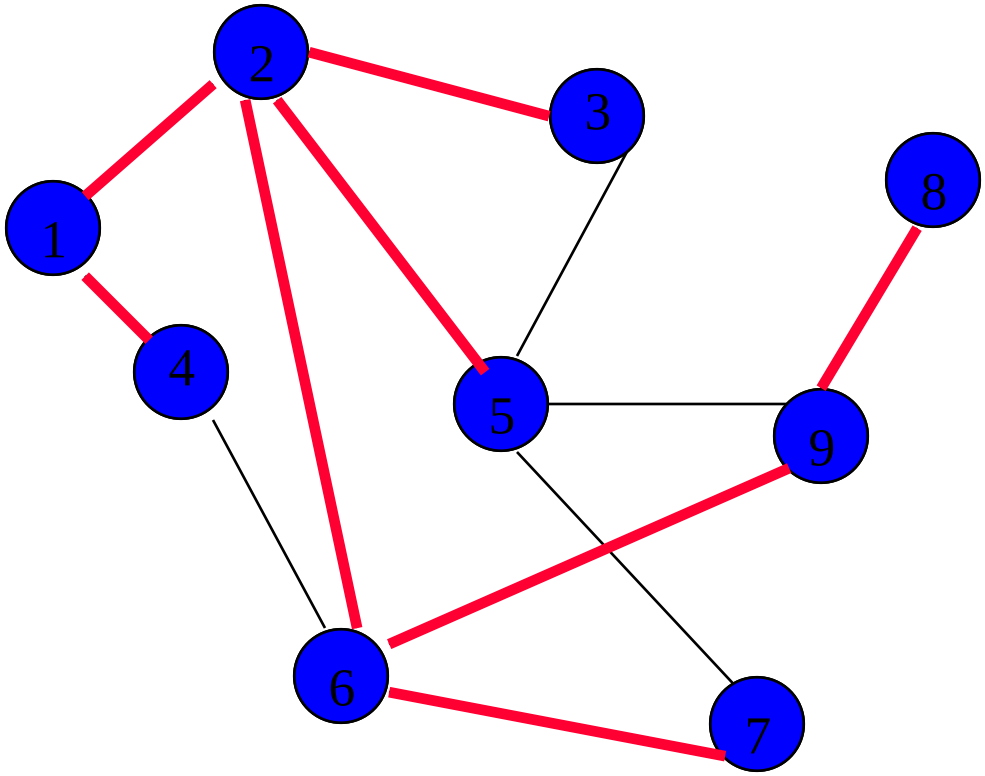
Connected Components



Time Complexity

- $O(n^2)$ when adjacency matrix used
- $O(n+e)$ when adjacency lists used (e is number of edges)

Spanning Tree



Breadth-first search from vertex 1.

Breadth-first spanning tree.

Spanning Tree

- Start a breadth-first search at any vertex of the graph.
- If graph is connected, the n-1 edges used to define a spanning tree (breadth-first spanning tree).
- Time
  - $O(n^2)$ when adjacency matrix used
  - $O(n+e)$ when adjacency lists used (e is number of edges)

Breadth-First Search Algo

- Traversal in nonempty graph beginning at a given vertex

```
getBreadthFirstSearch(originVertex)
  Mark originVertex as visited
  WorkingQueue.enqueue(originVertex)
  while (! WorkingQueue.isEmpty())
  {
     frontVertex = WorkingQueue.dequeue()
     while (frontVertex has an unvisited neighbor)
     {    nextNeighbor = next unvisited neighbor of frontVertex
          Mark nextNeighbor as visited
          WorkingQueue.enqueue(nextNeighbor)
      }
 }
```