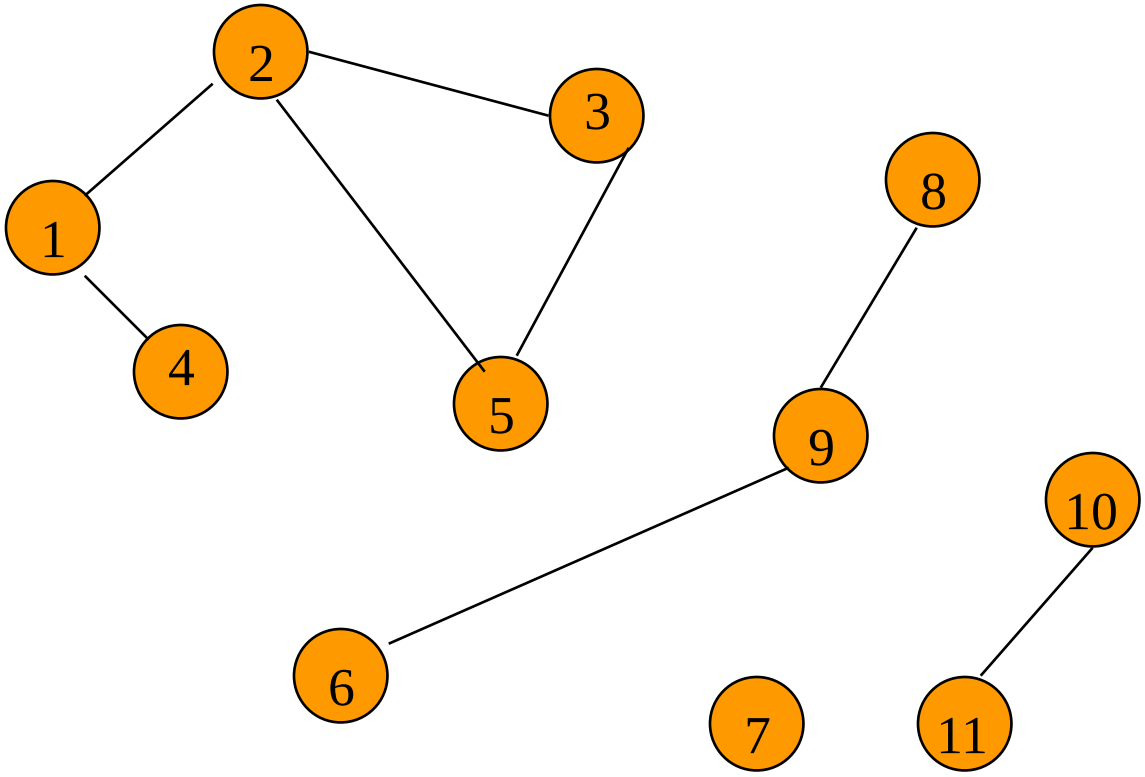
Graph Search Methods

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

Graph Search Methods

- In both DFS and BFS, the nodes of the undirected graph are visited in a systematic manner so that every node is visited exactly one.

- Both BFS and DFS give rise to a tree:
  - When a node x is visited, it is labeled as visited, and it is added to the tree
  - If the traversal got to node x from node y, y is viewed as the parent of x, and x a child of y

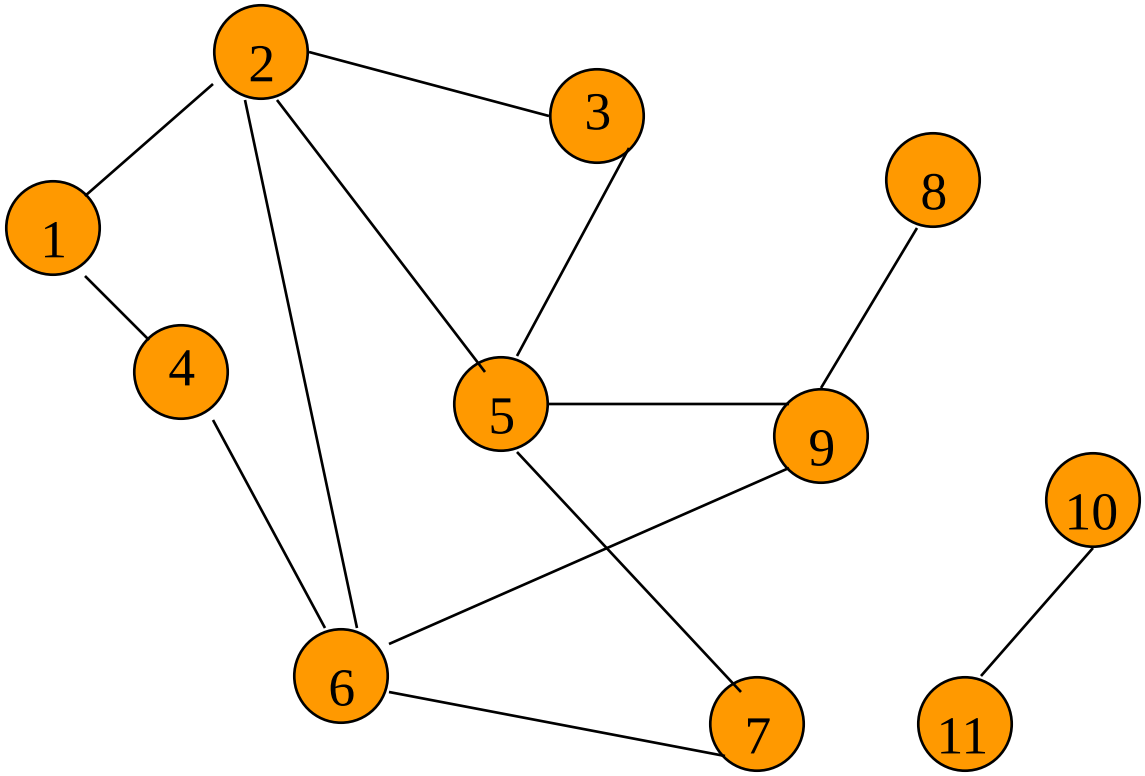
Breadth-First Search

- Visit start vertex and put into a queue.

- Repeatedly remove a vertex from the queue, visit its unvisited adjacent vertices, put newly visited vertices into the queue

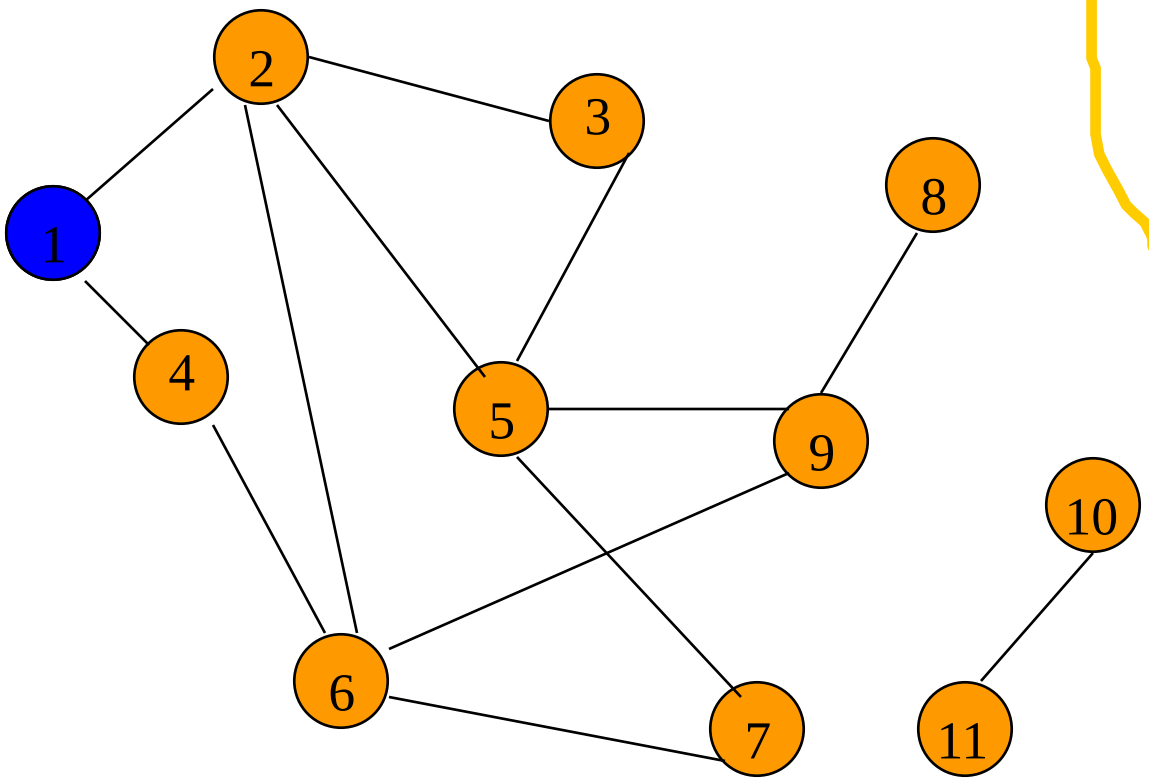
  until the queue is empty.

Breadth-First Search Example



Start search at vertex 1.

Breadth-First Search Example

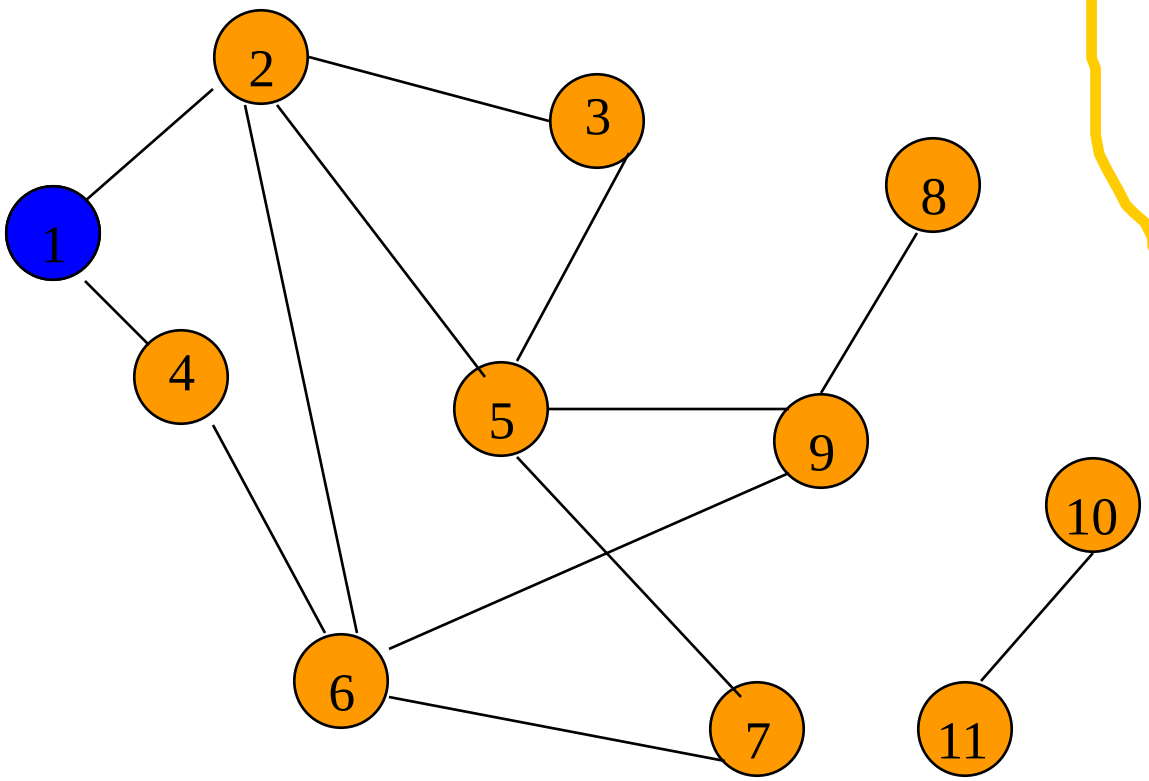


Queue

1

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

Breadth-First Search Example

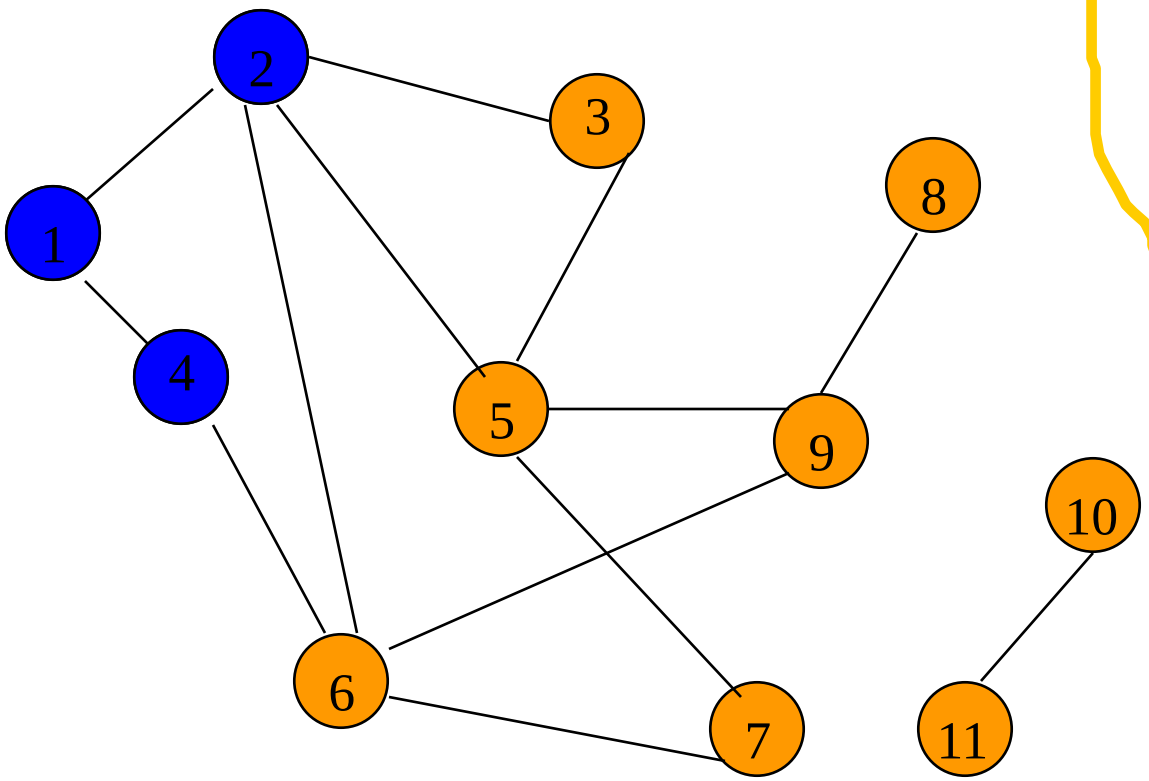


Queue

1

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

Breadth-First Search Example

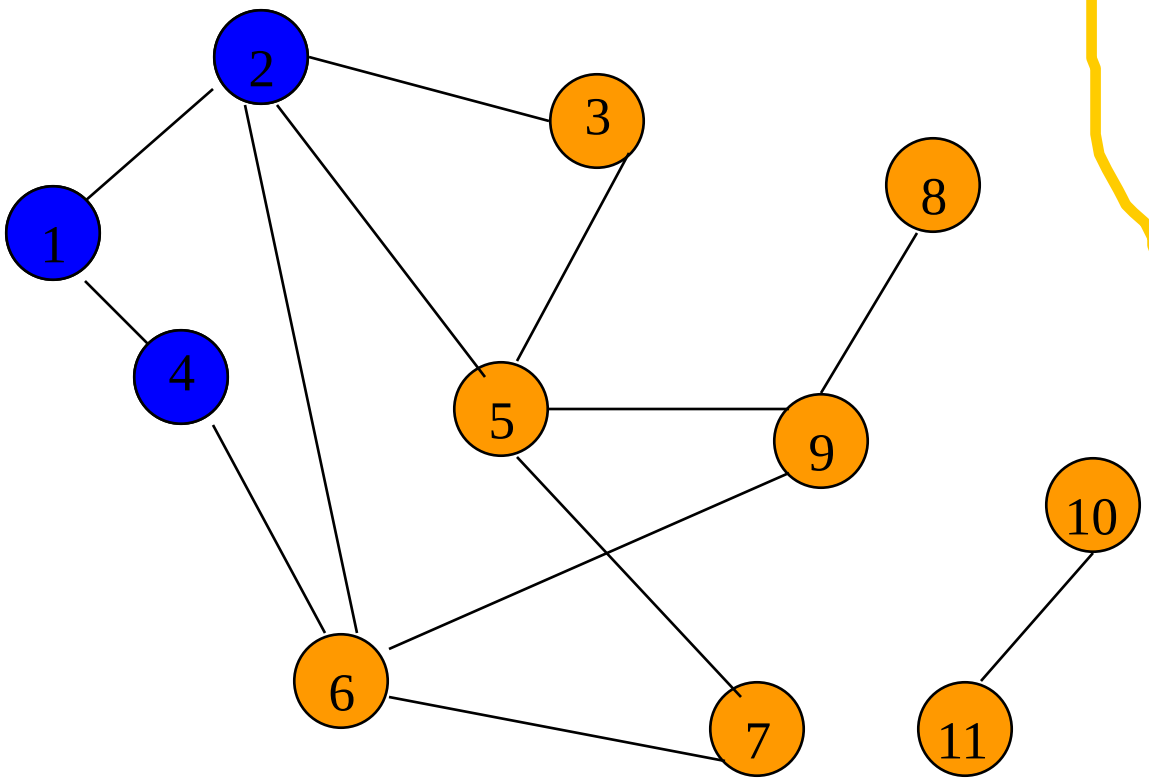


Queue

2 4

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

Breadth-First Search Example

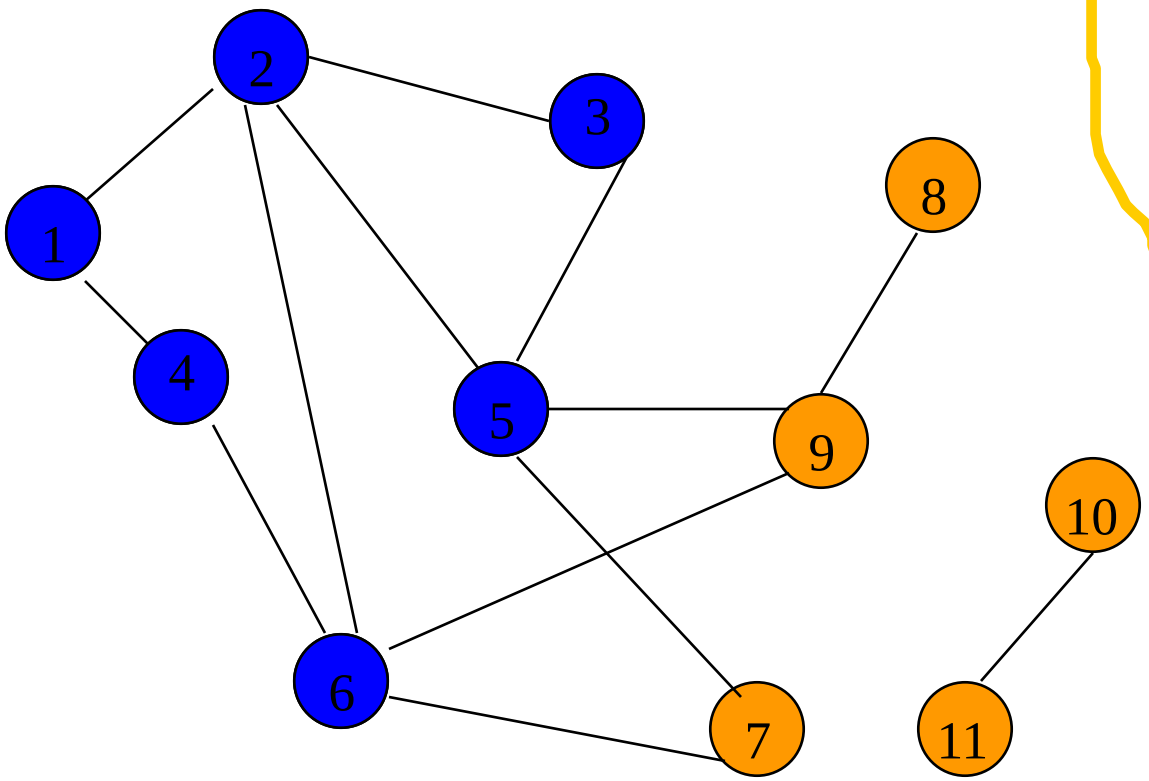


Queue

2 4

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

Breadth-First Search Example

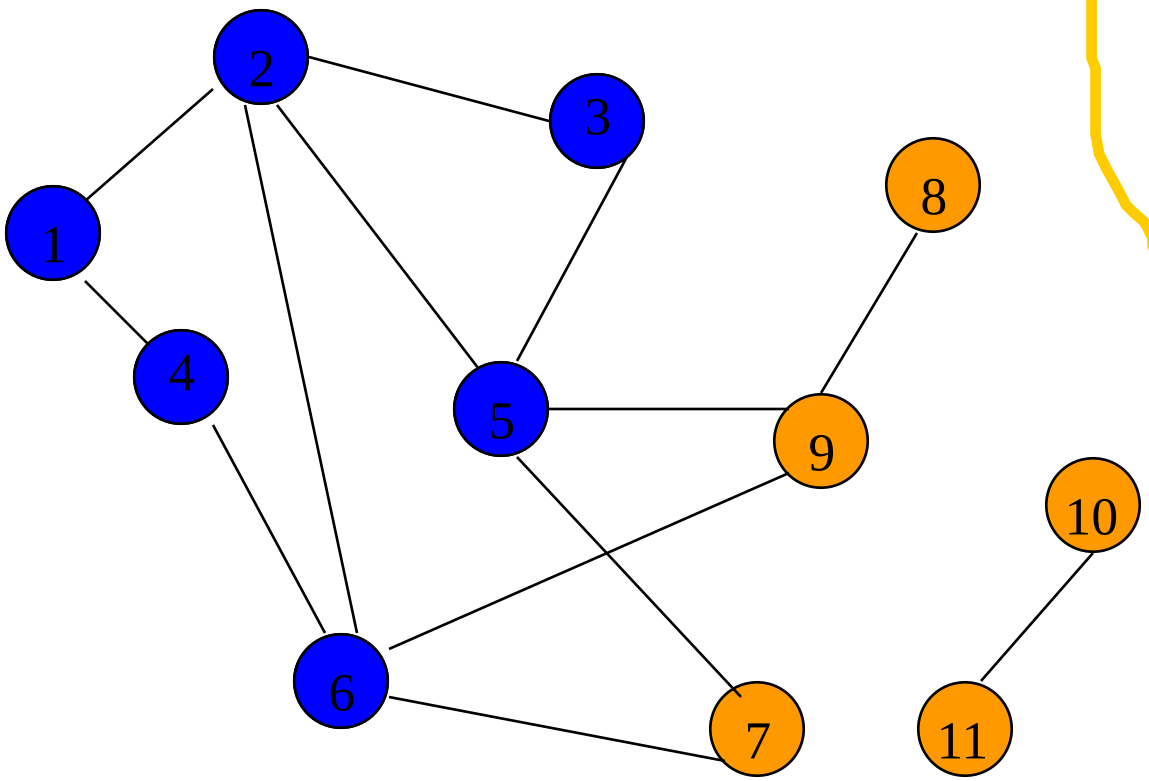


Queue

4 5 3 6

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

Breadth-First Search Example

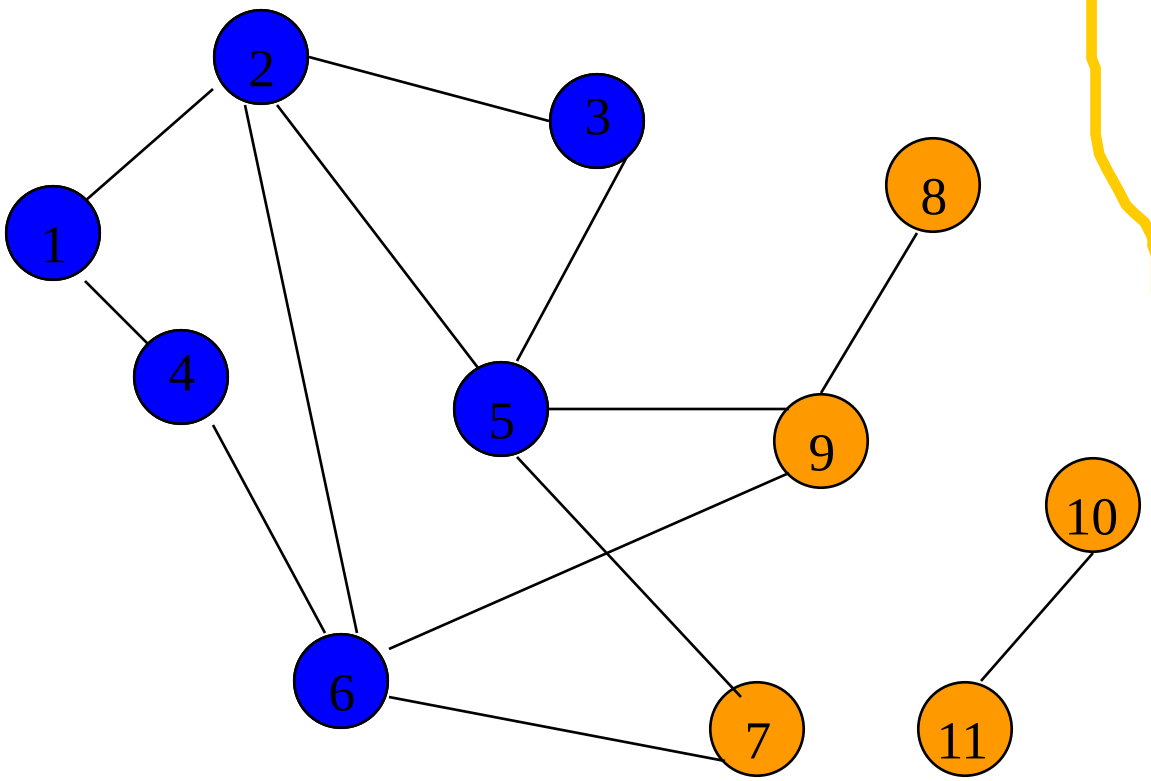


Queue

4 5 3 6

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

Breadth-First Search Example

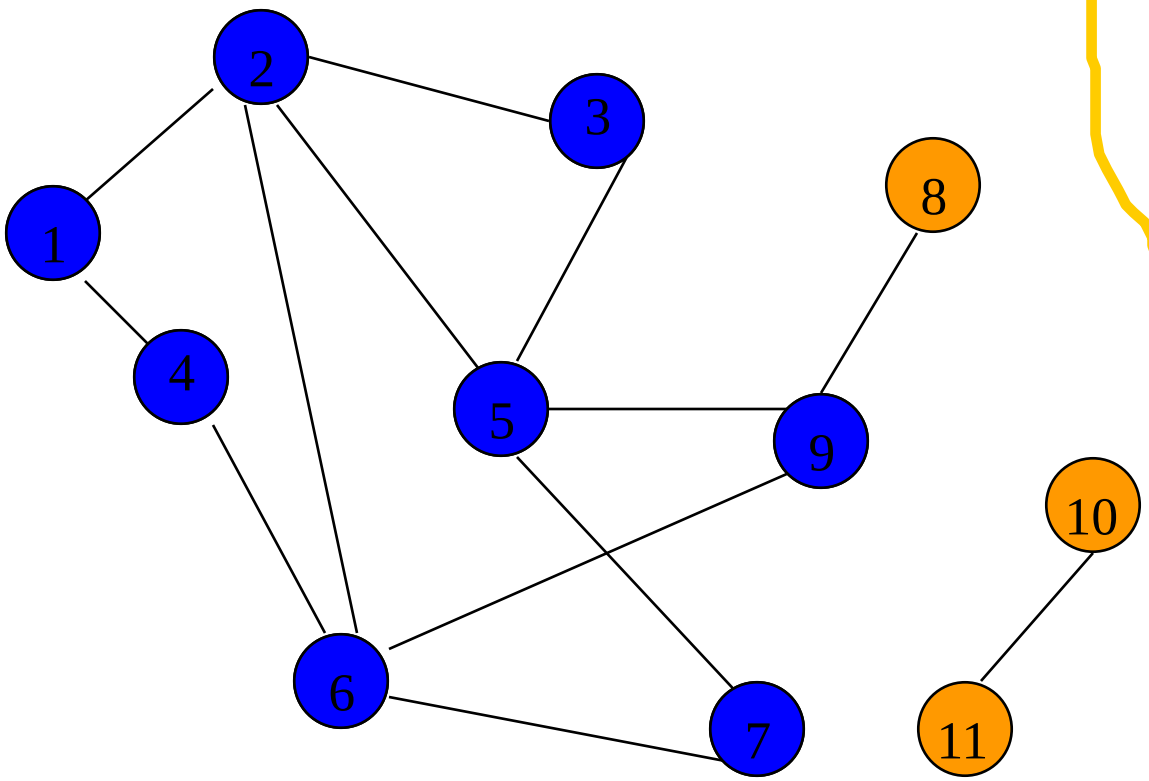


Queue

5 3 6

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

Breadth-First Search Example

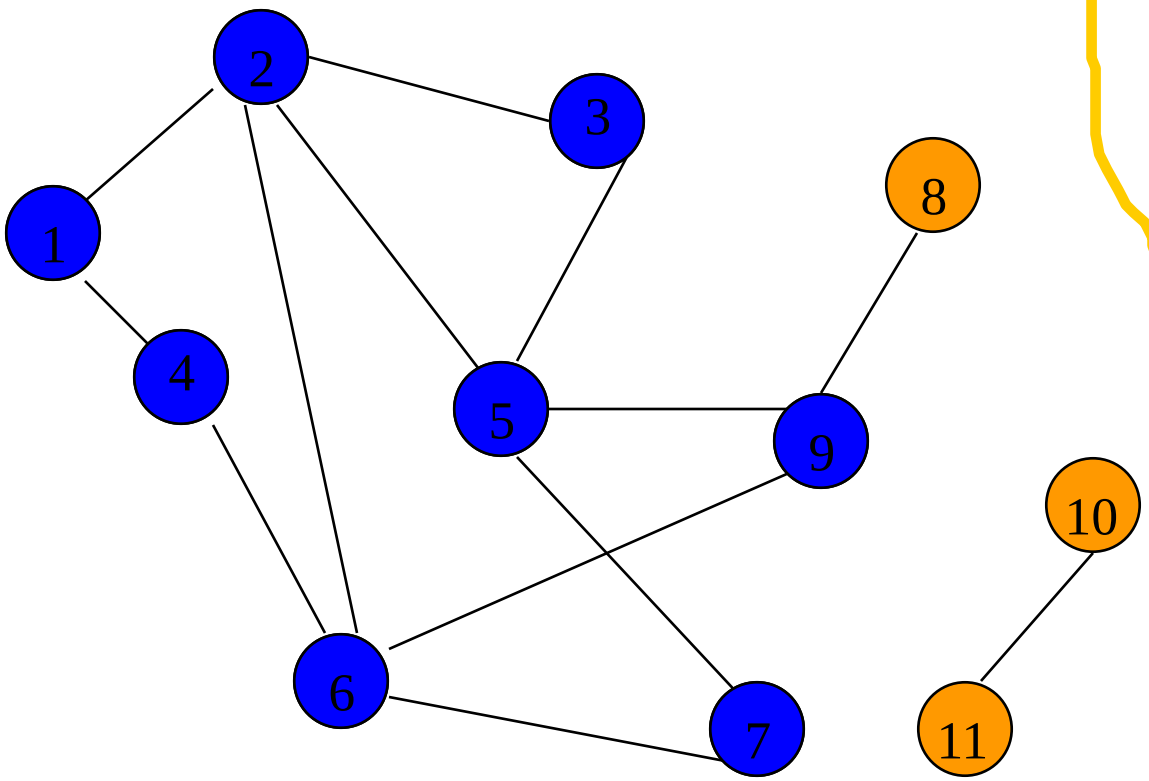


Queue

3 6 9 7

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

Breadth-First Search Example

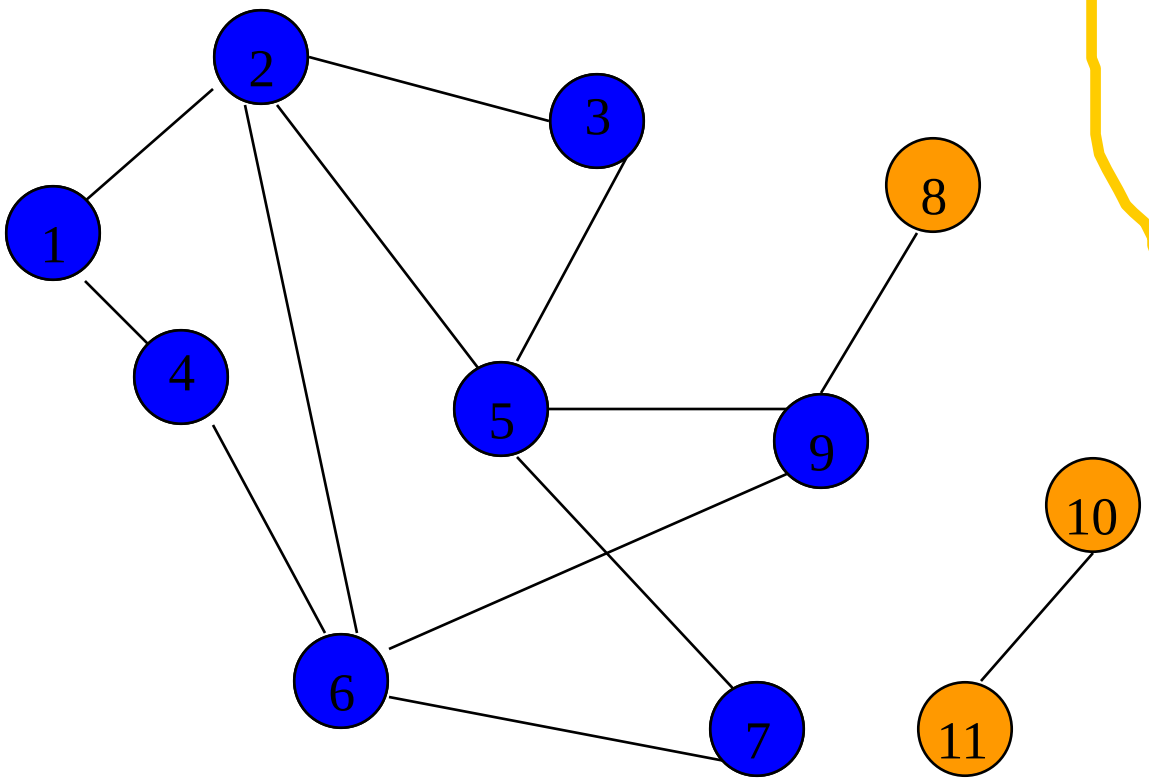


Queue

3 6 9 7

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

Breadth-First Search Example



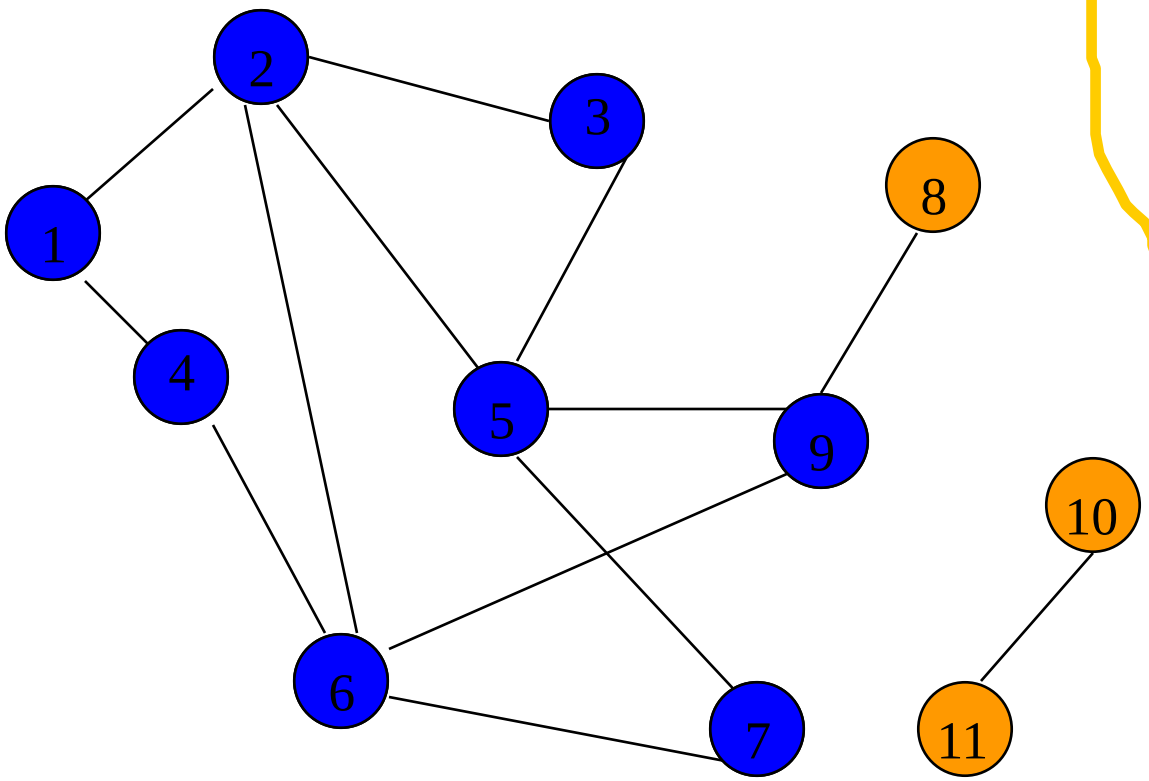
Queue

6 9 7

Remove 6 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example

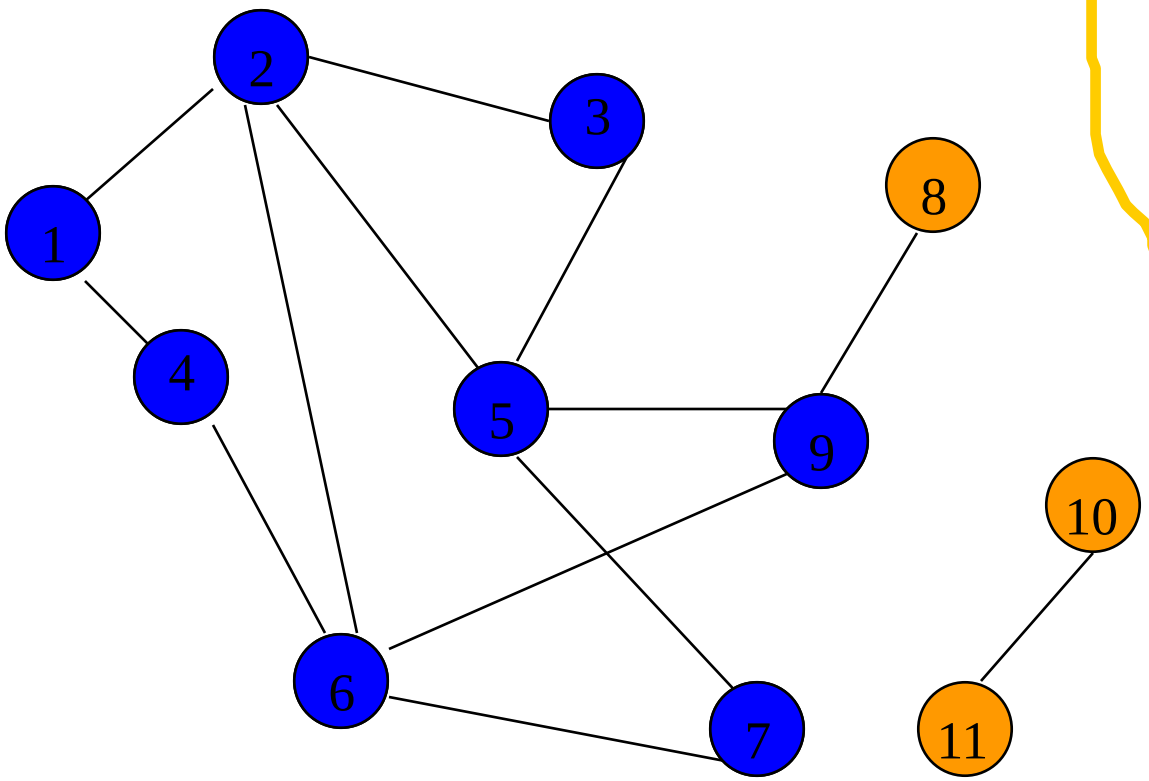


Queue

9 7

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

Breadth-First Search Example



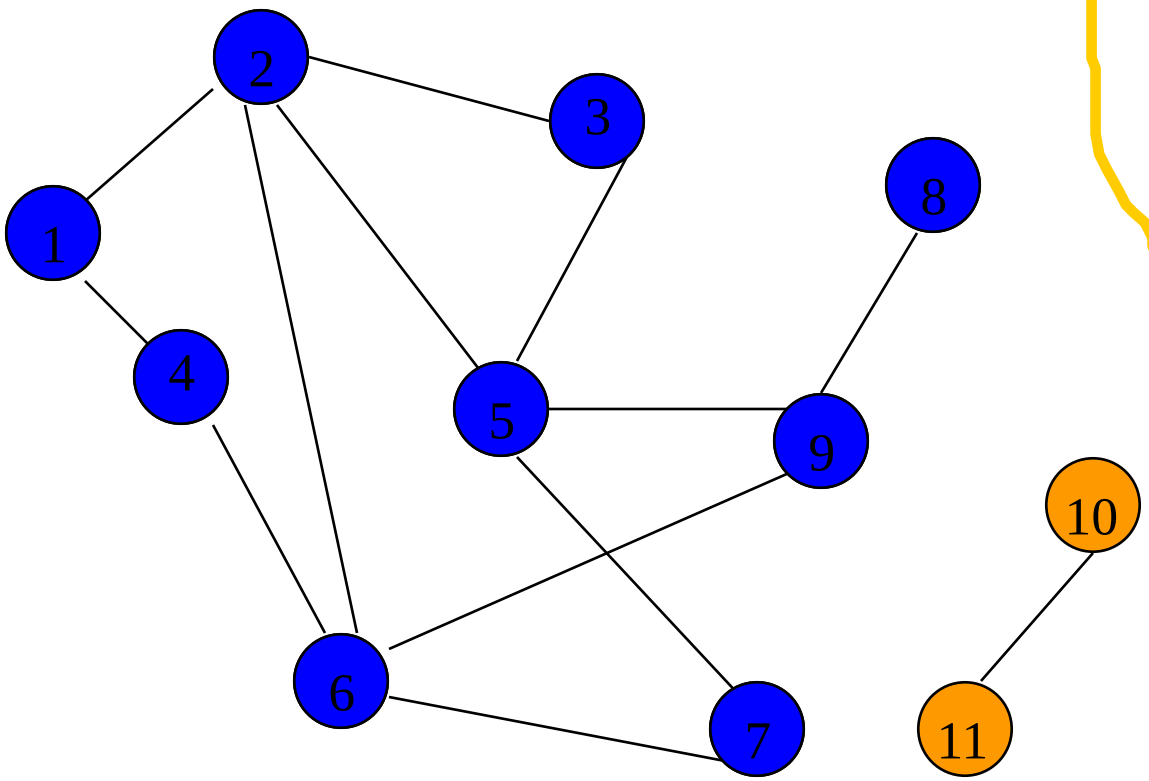
Queue

9 7

Remove 9 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example

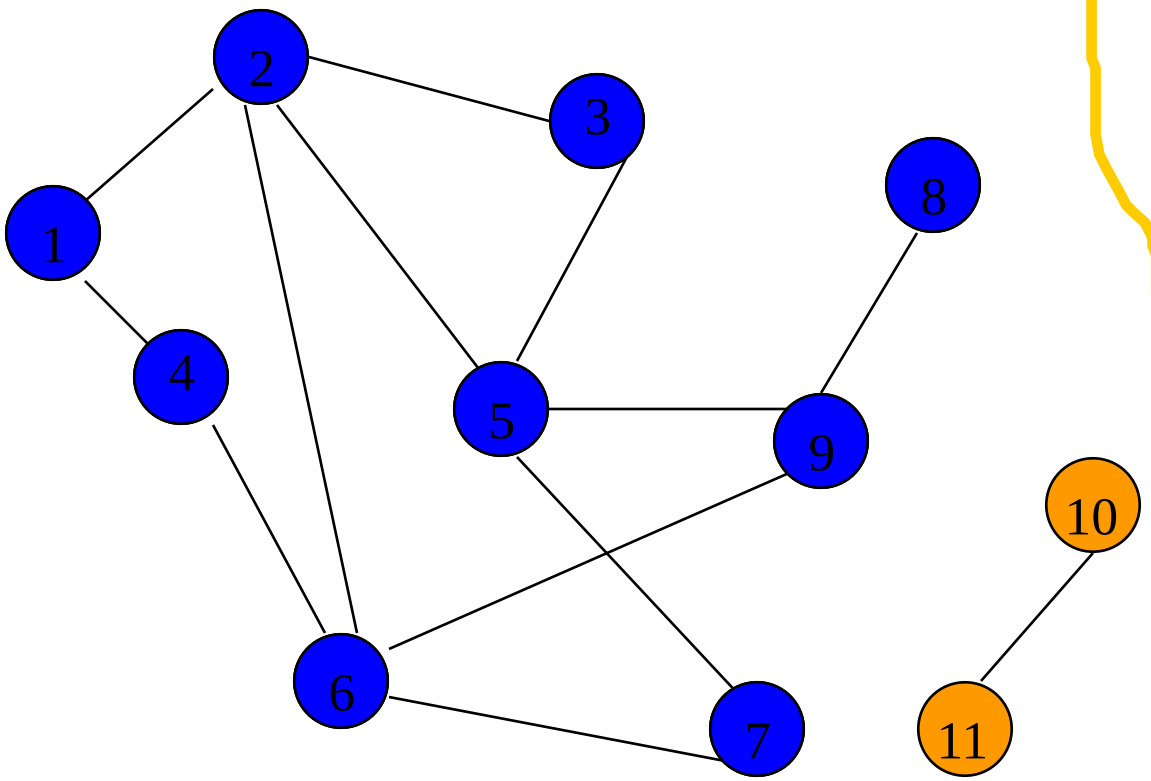


Queue

7 8

Remove 9 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example



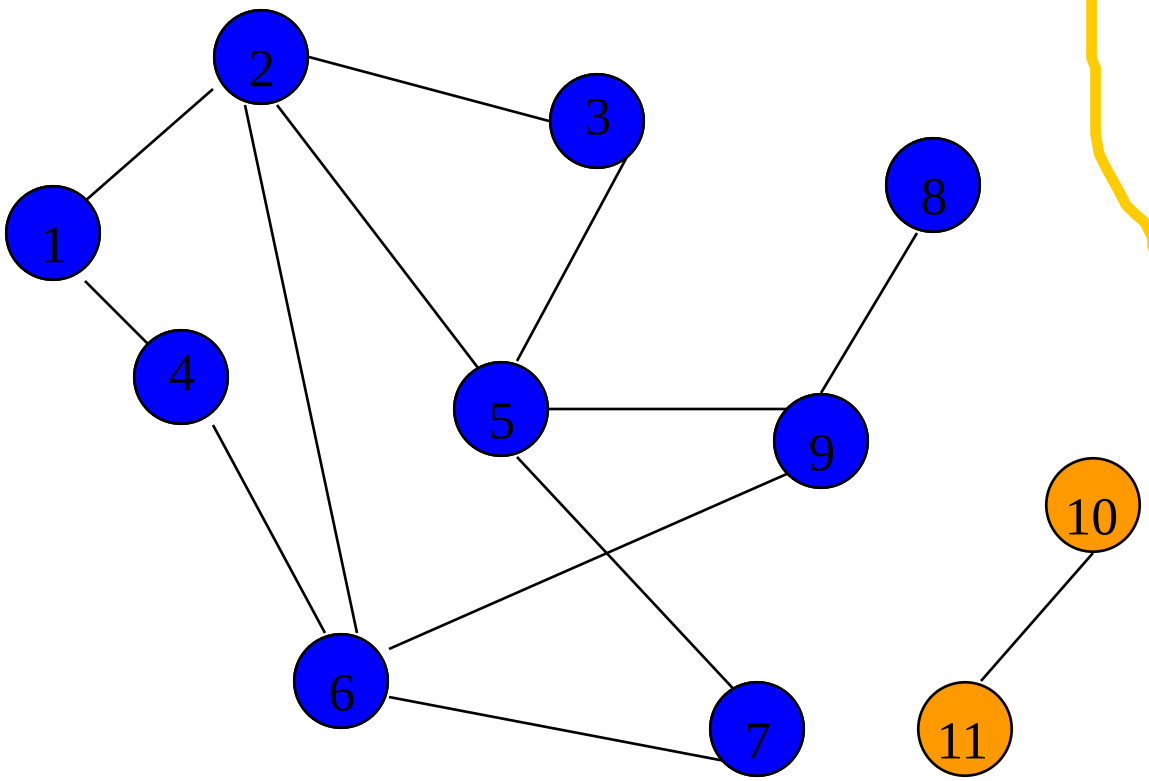
Queue

8

Remove 7 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example



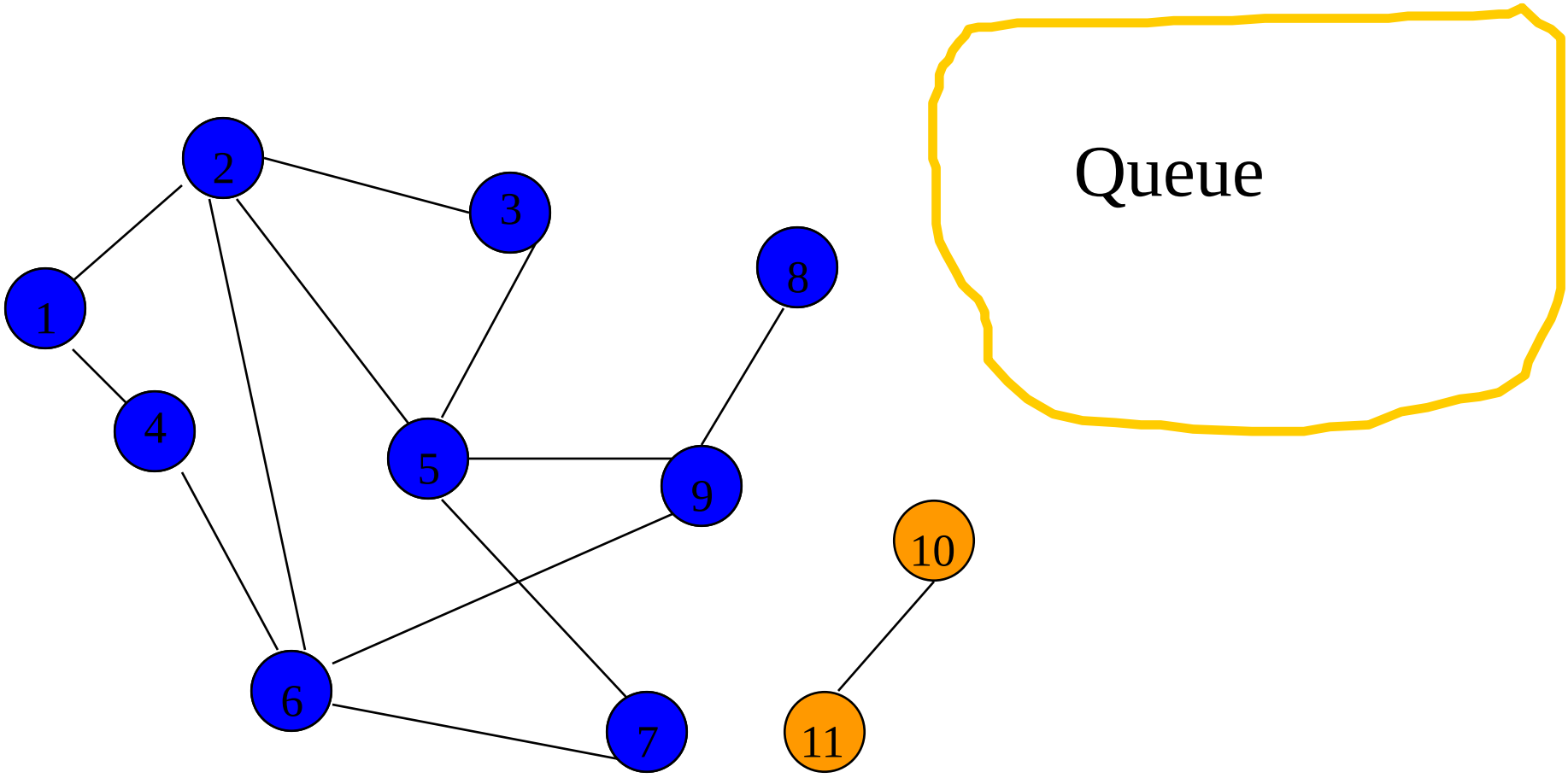
Queue

8

Remove 8 from Q; visit adjacent unvisited vertices;

put in Q.

Breadth-First Search Example



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 removed from) the queue exactly once.
- When a vertex is removed from the queue, we examine its adjacent vertices.
  - O(vertex degree) if adjacency lists are used
- Total time, when adjacency lists are used:

  O(n + sum of degrees of the vertices in the component)

  = O(n + number of edges in the component)

  = O(n + m) if the graph is connected.

Depth-First Search

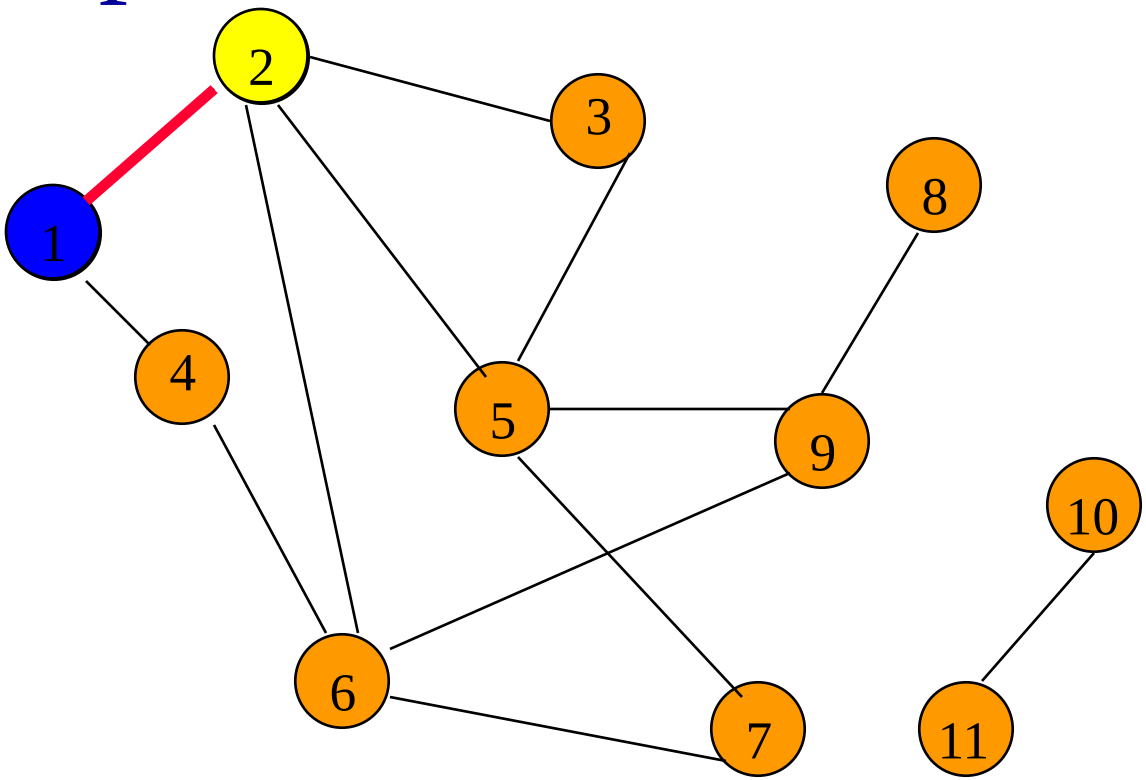
- DFS follows the following rules:
  1. Select an unvisited node x, visit it, and treat as the current node
  2. Find an unvisited neighbor of the current node, visit it, and make it the new current node;
  3. If the current node has no unvisited neighbors, backtrack to the its parent, and make that parent the new current node;
  4. Repeat steps 3 and 4 until no more nodes can be visited.
  5. If there are still unvisited nodes, repeat from step 1.

Depth-First Search

pseudo code

```
depthFirstSearch(v)
{
   Label vertex v as reached.
   for (each unreached vertex u adjacent to v)
      depthFirstSearch(u);
}
```

Depth-First Search Example

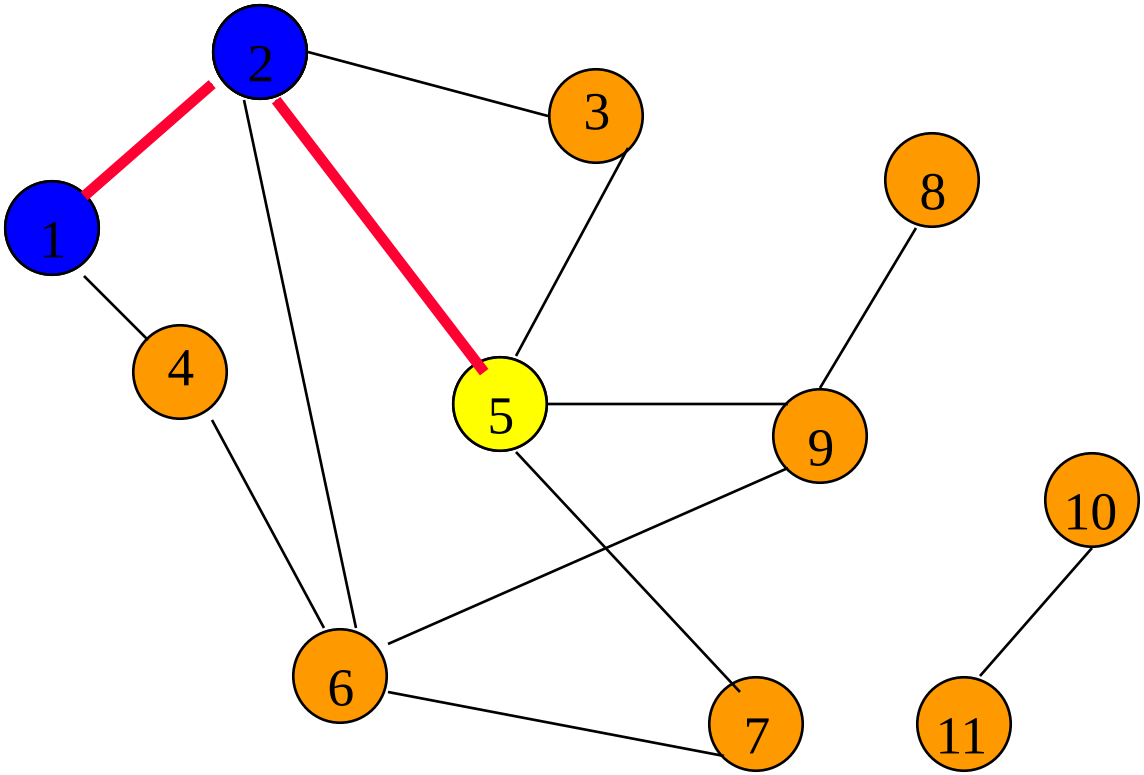


Start search at vertex 1.

Label vertex 1 and do a depth first search from either 2 or 4.

Suppose that vertex 2 is selected.

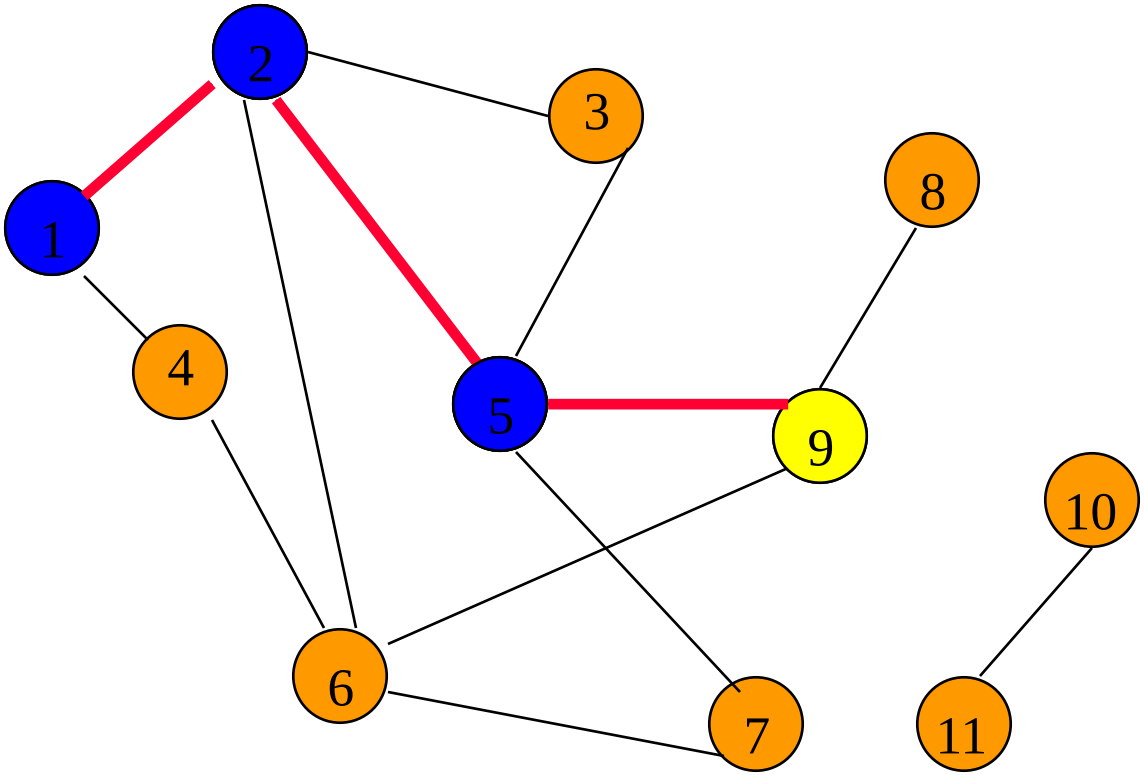
Depth-First Search Example



Label vertex 2 and do a depth first search from either 3, 5, or 6.

Suppose that vertex 5 is selected.

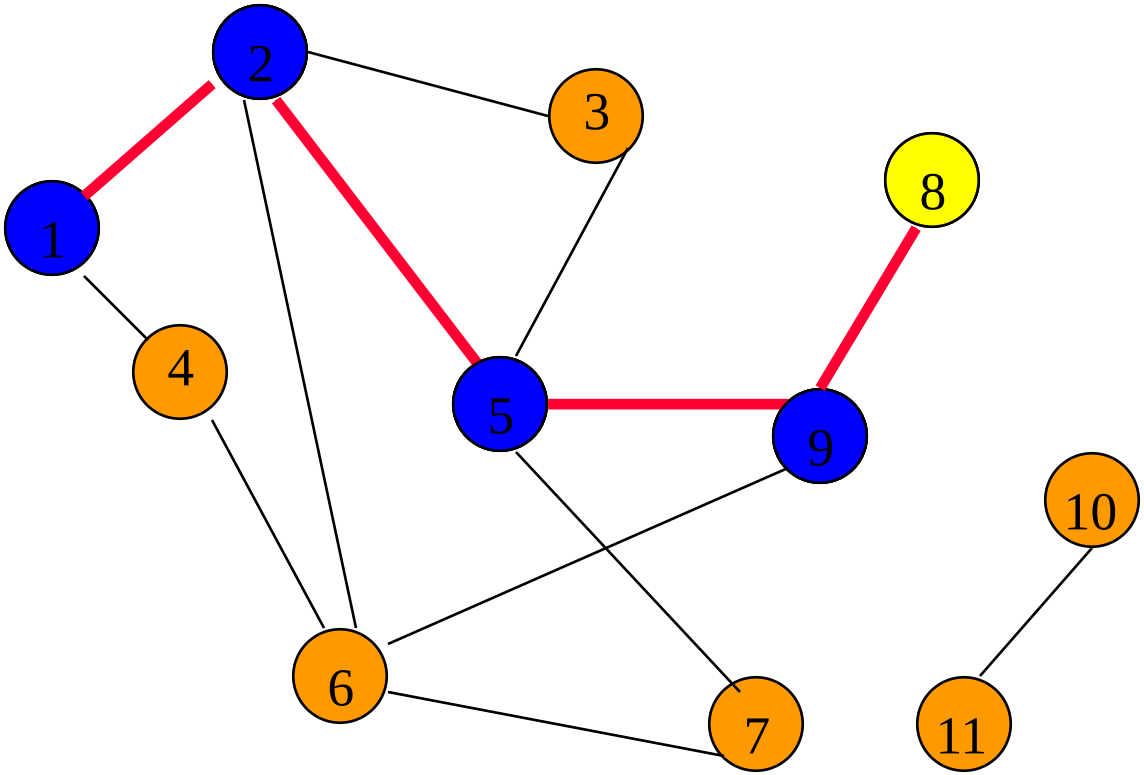
Depth-First Search Example



Label vertex 5 and do a depth first search from either 3, 7, or 9.

Suppose that vertex 9 is selected.

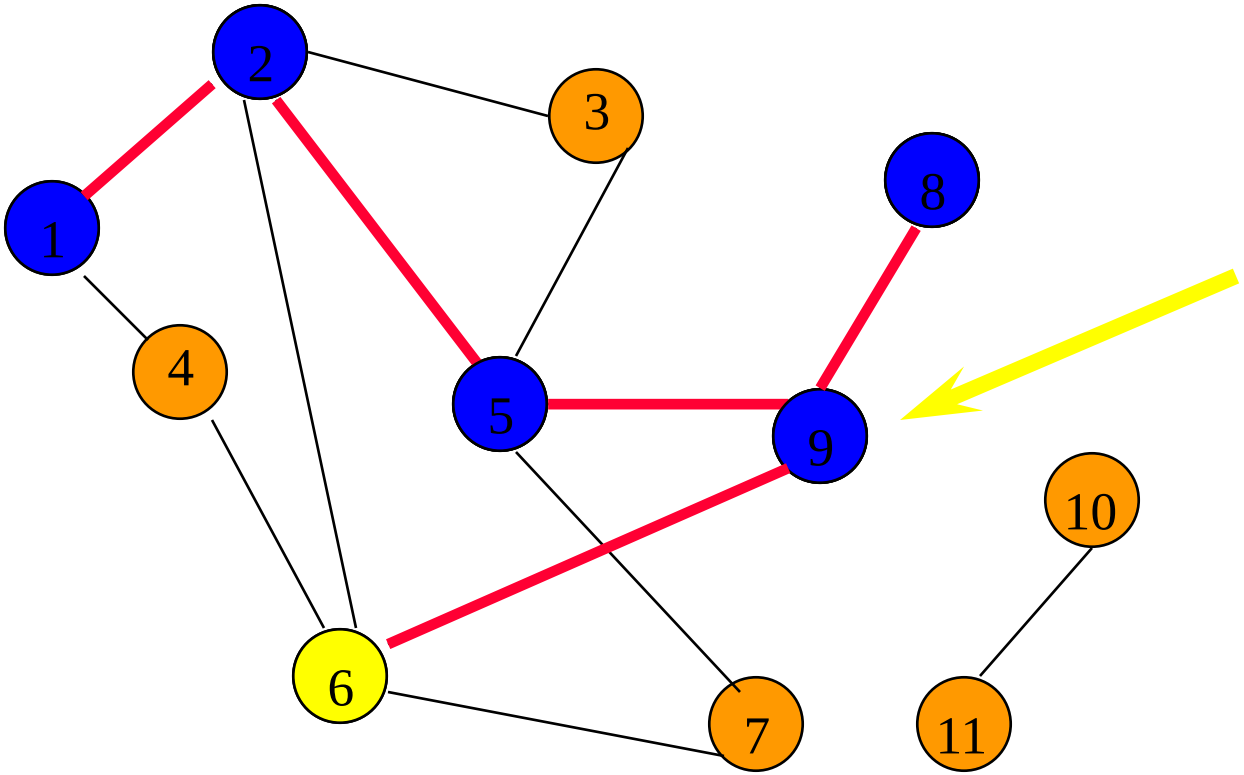
Depth-First Search Example



Label vertex 9 and do a depth first search from either 6 or 8.

Suppose that vertex 8 is selected.

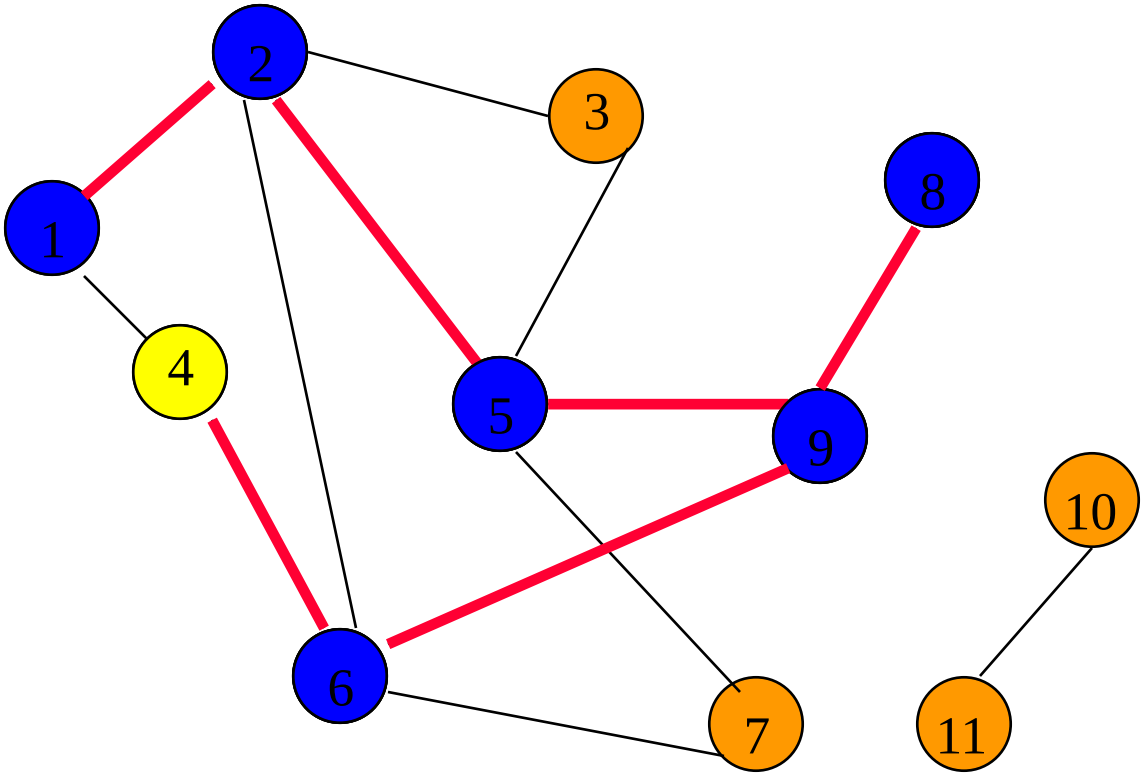
Depth-First Search Example



Label vertex 8 and return to vertex 9.

From vertex 9 do a dfs(6).

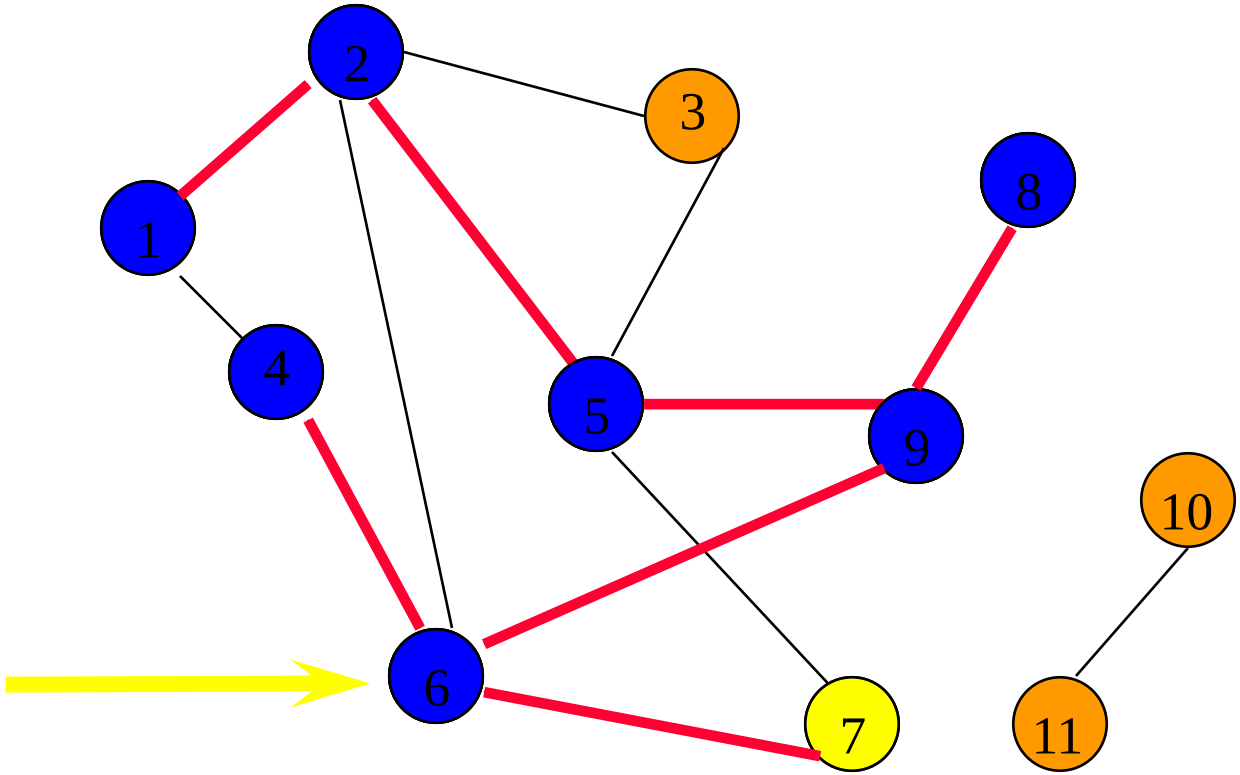
Depth-First Search Example



Label vertex 6 and do a depth first search from either 4 or 7.

Suppose that vertex 4 is selected.

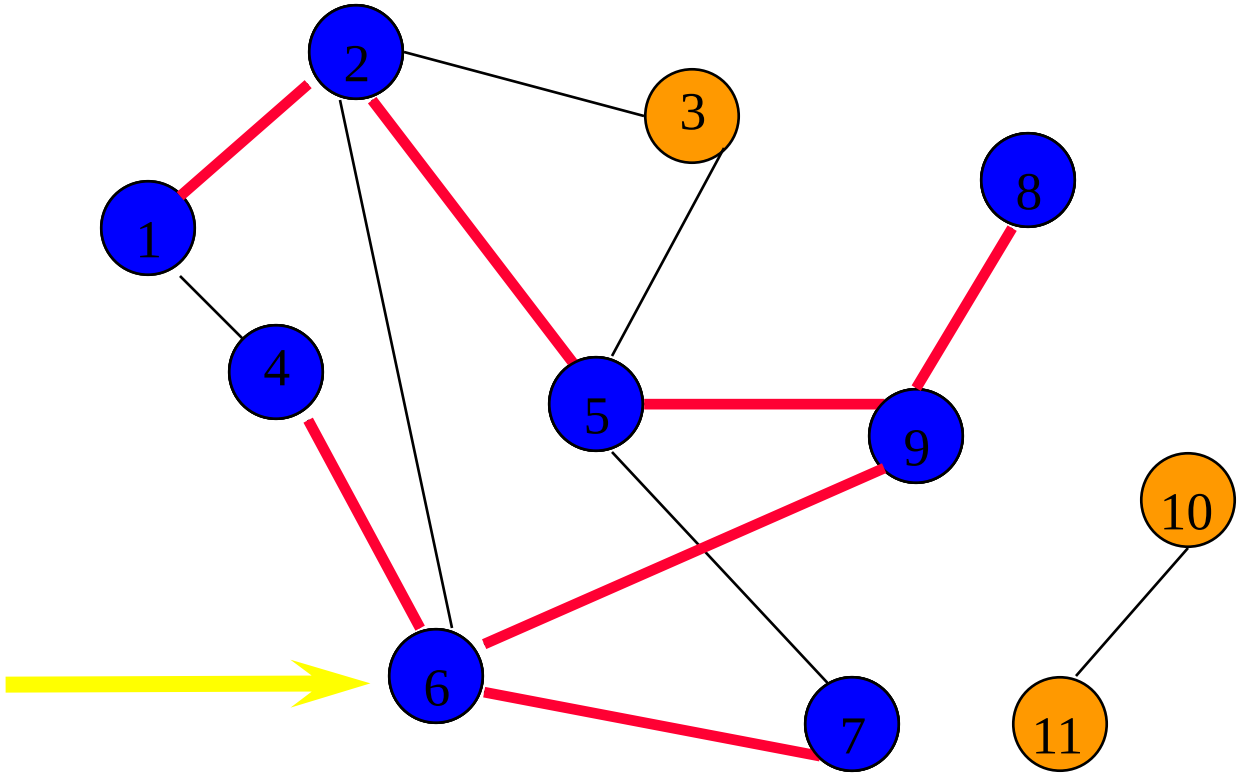
Depth-First Search Example



Label vertex 4 and return to 6.

From vertex 6 do a dfs(7).

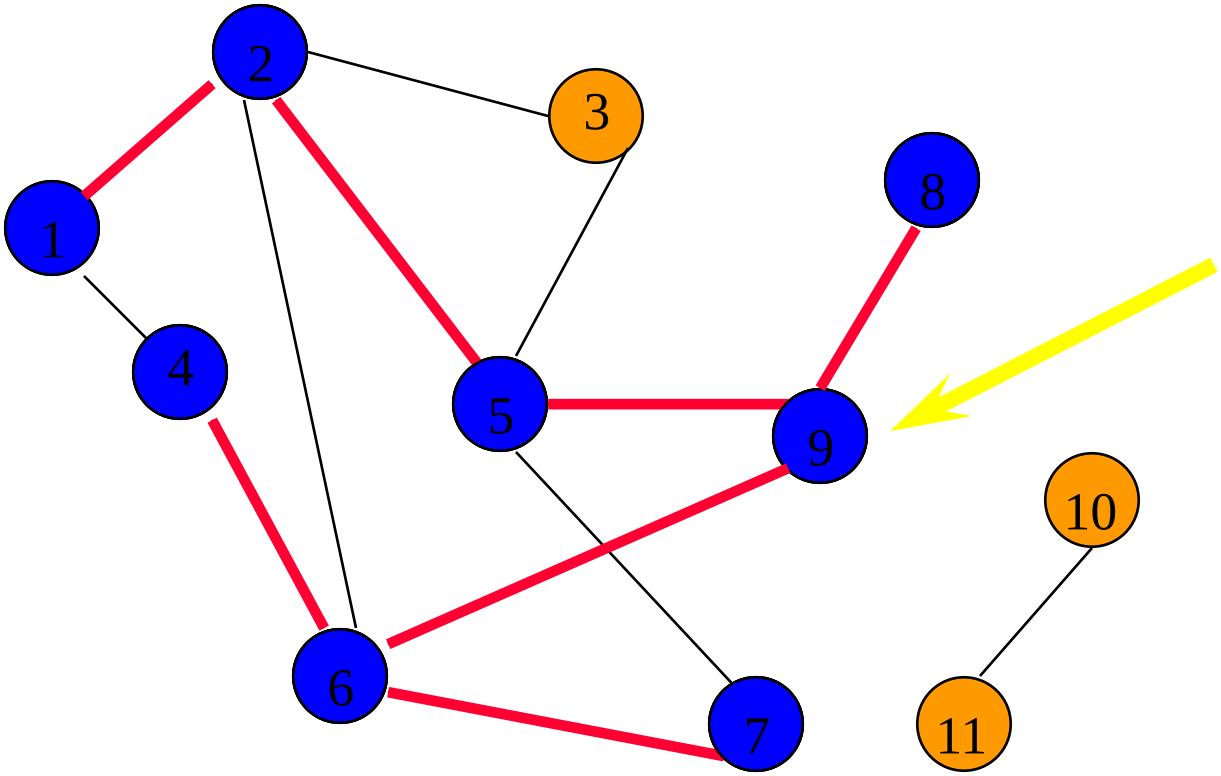
Depth-First Search Example



Label vertex 7 and return to 6.

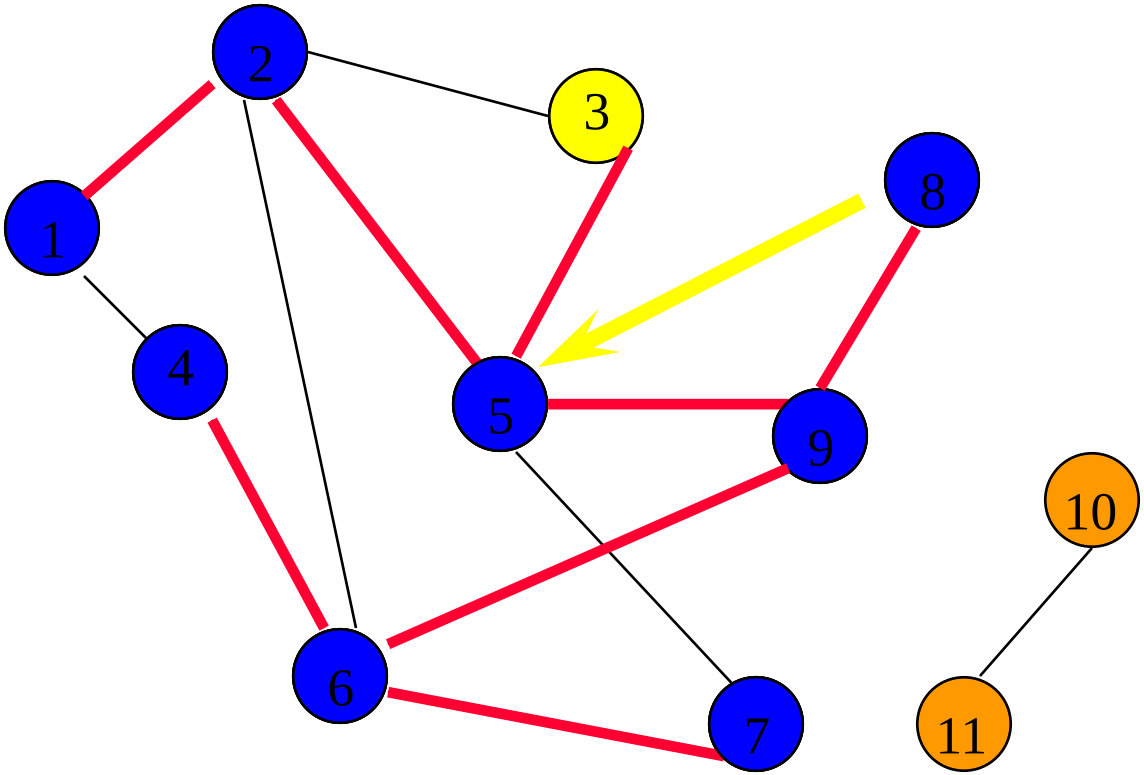
Return to 9.

Depth-First Search Example



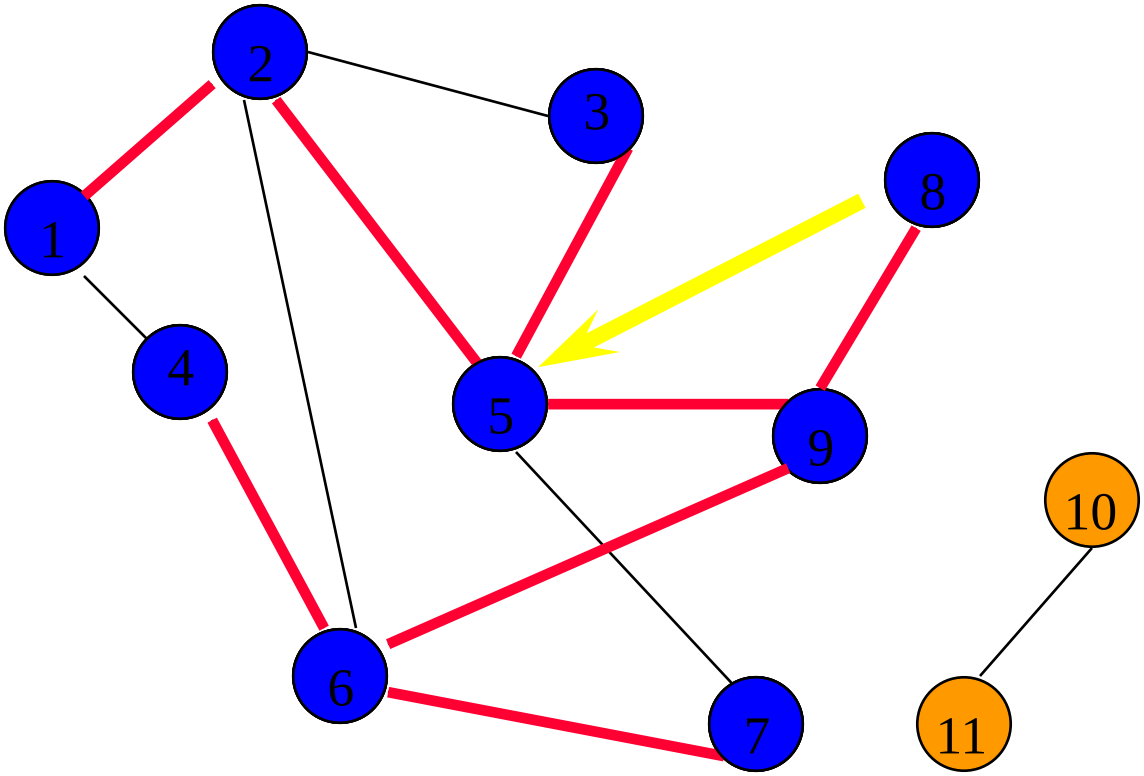
Return to 5.

Depth-First Search Example



Do a dfs(3).

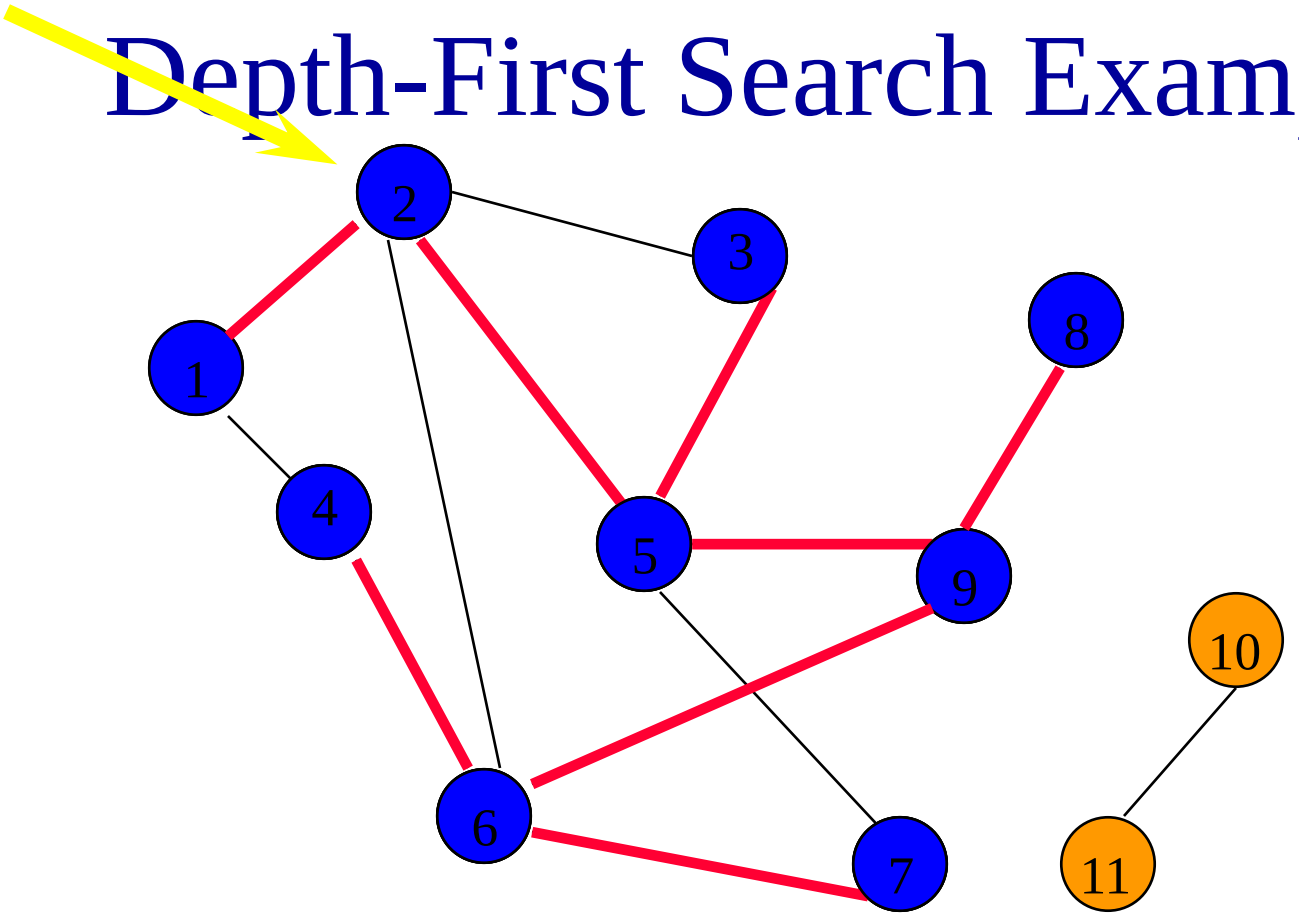
Depth-First Search Example



Label 3 and return to 5.

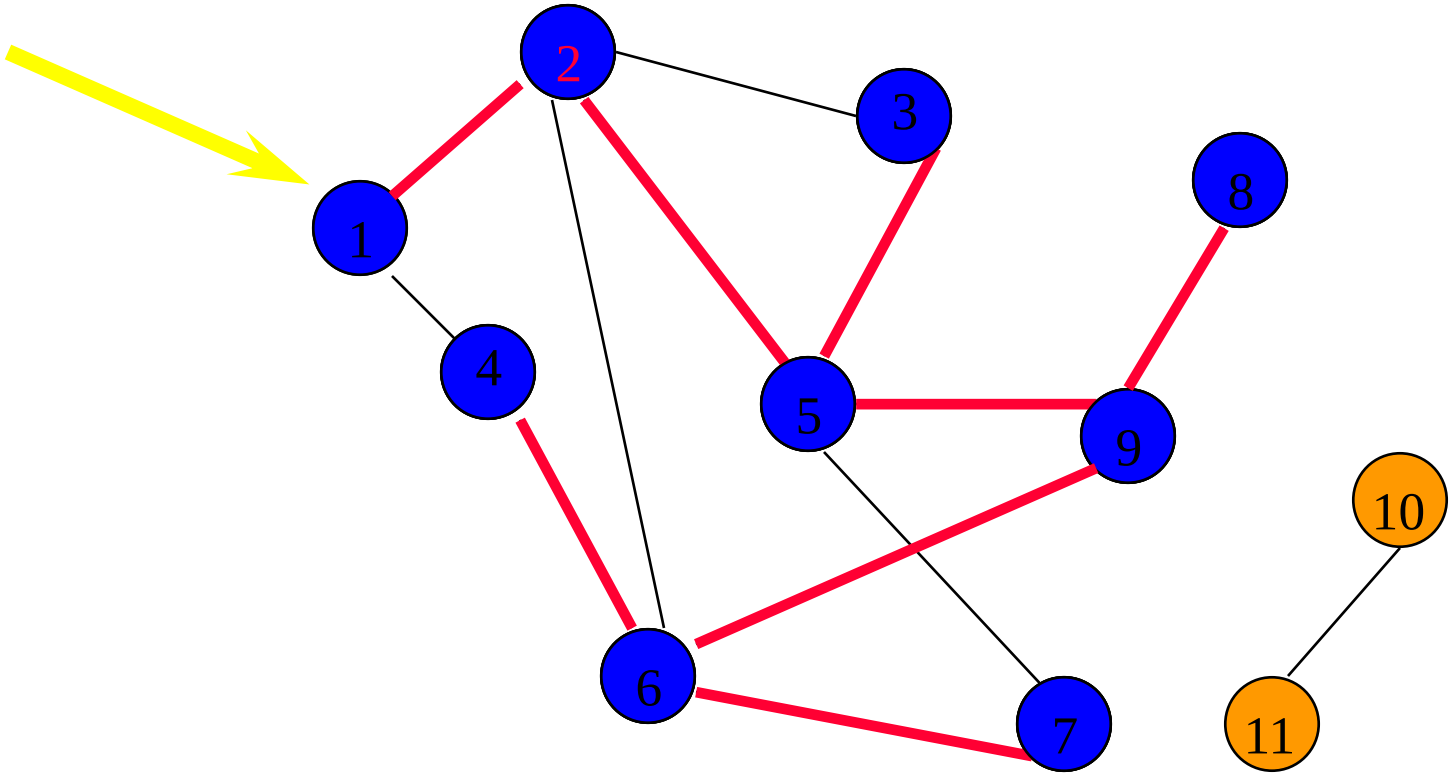
Return to 2.

Depth-First Search Example



Return to 1.

Depth-First Search Example



Return to invoking method.

Depth-First Search Properties

- Same complexity as BFS.
- Same properties with respect to path finding, connected components, and spanning trees.
- Edges used to reach unlabeled vertices define a depth-first spanning tree when the graph is connected.
- There are problems for which bfs is better than dfs and vice versa.