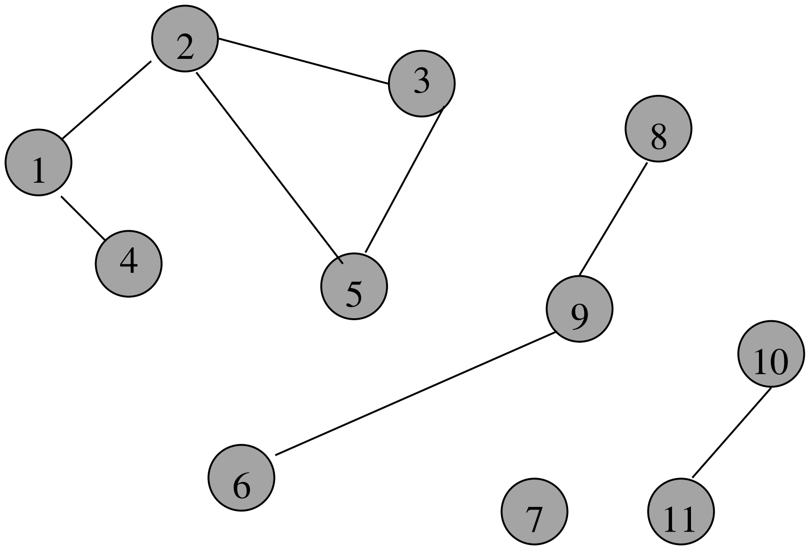
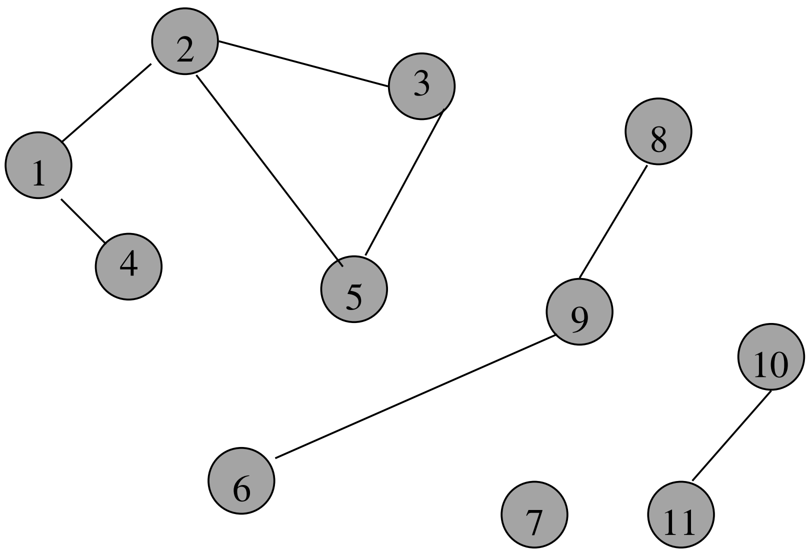
Graph Search Methods

- 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.



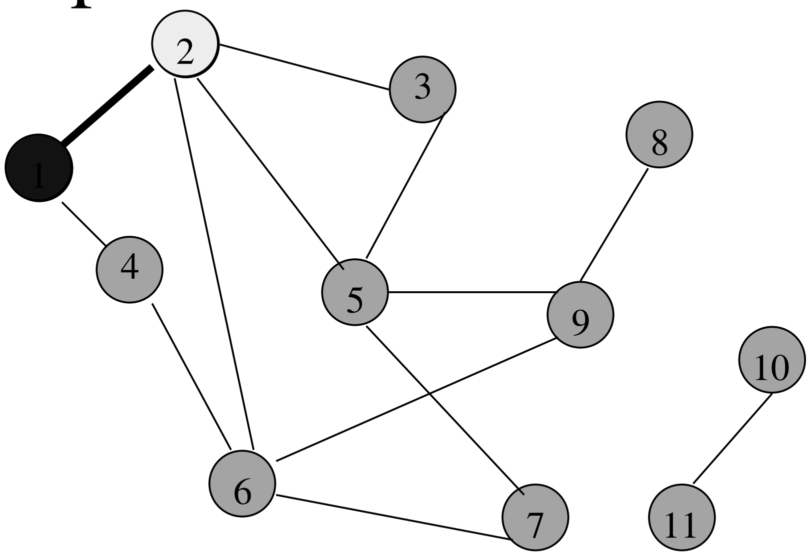
Graph Search Methods

- 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:
  - Depth-first search.
  - Breadth-first search.

Depth-First Search

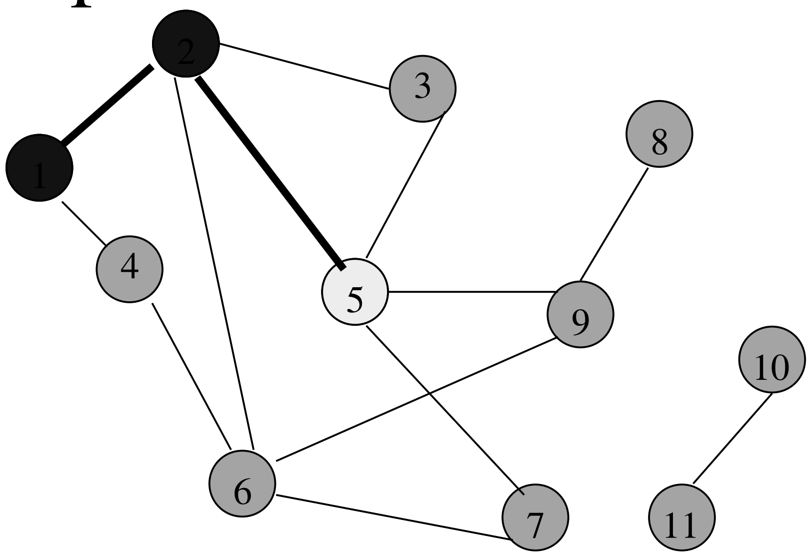
```
DFS(v)
{
  Label vertex v as reached.
  for (each unreached vertex u
                          adjacenct from v)
    DFS(u);
}
```

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.

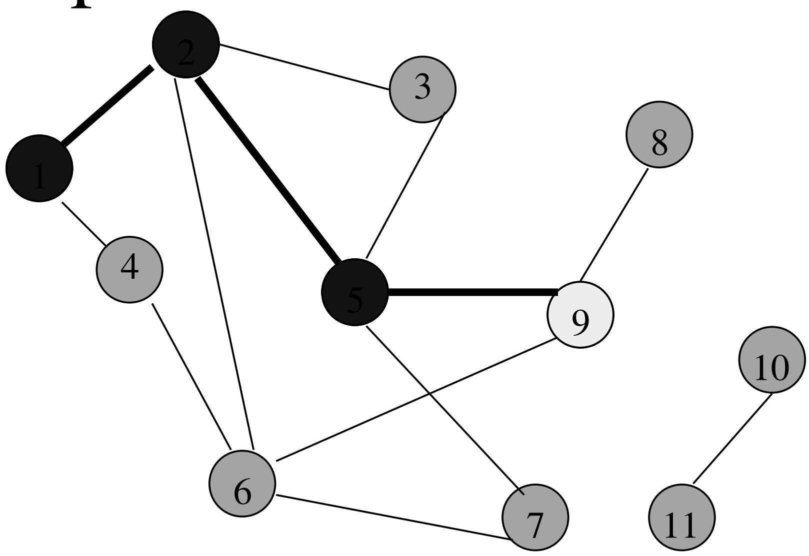
Depth-First Search Example



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

Suppose that vertex 5 is selected.

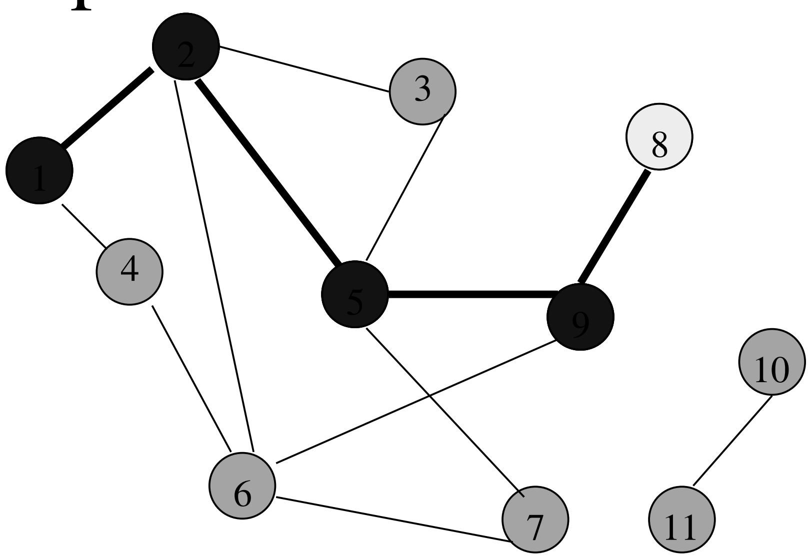
Depth-First Search Example



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

Suppose that vertex 9 is selected.

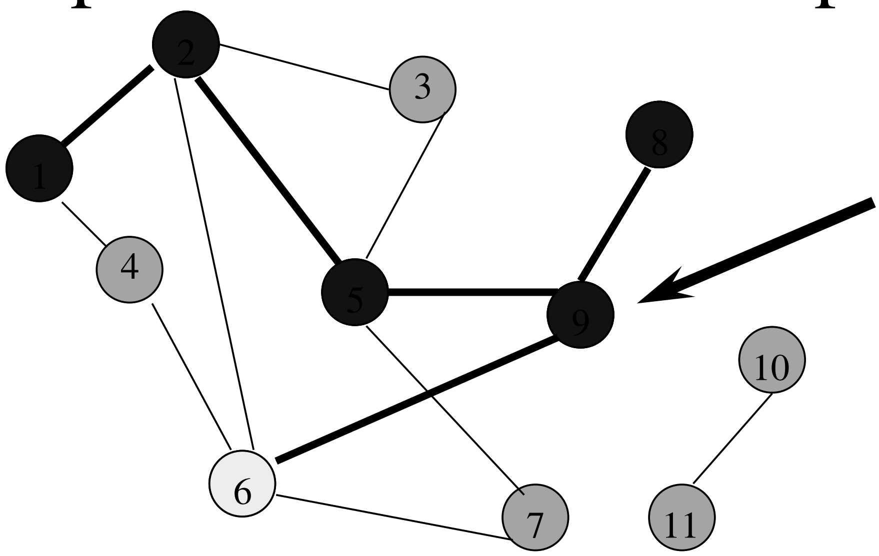
Depth-First Search Example



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

Suppose that vertex 8 is selected.

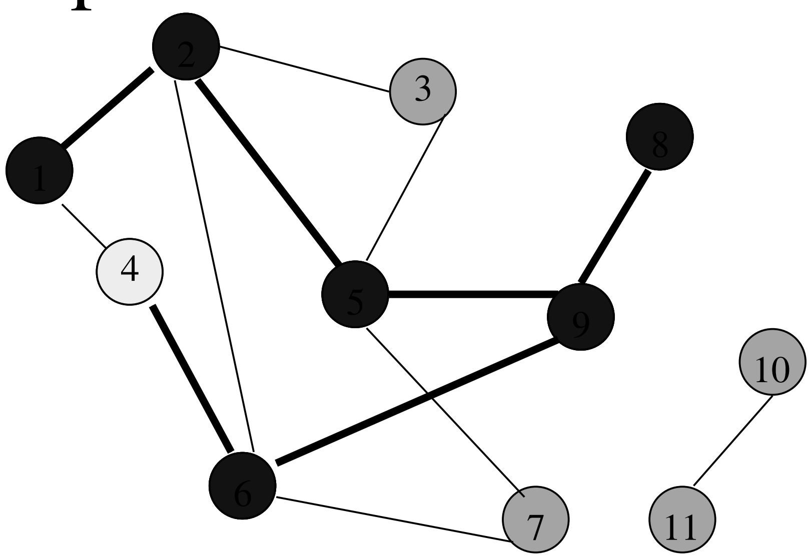
Depth-First Search Example



Label vertex 8 and return to vertex 9.

From vertex 9 do a DFS(6).

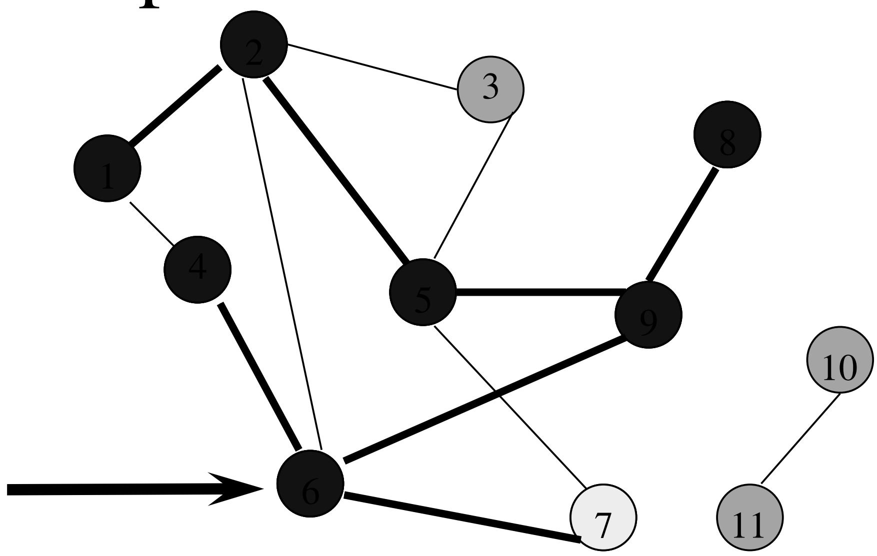
Depth-First Search Example



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

Suppose that vertex 4 is selected.

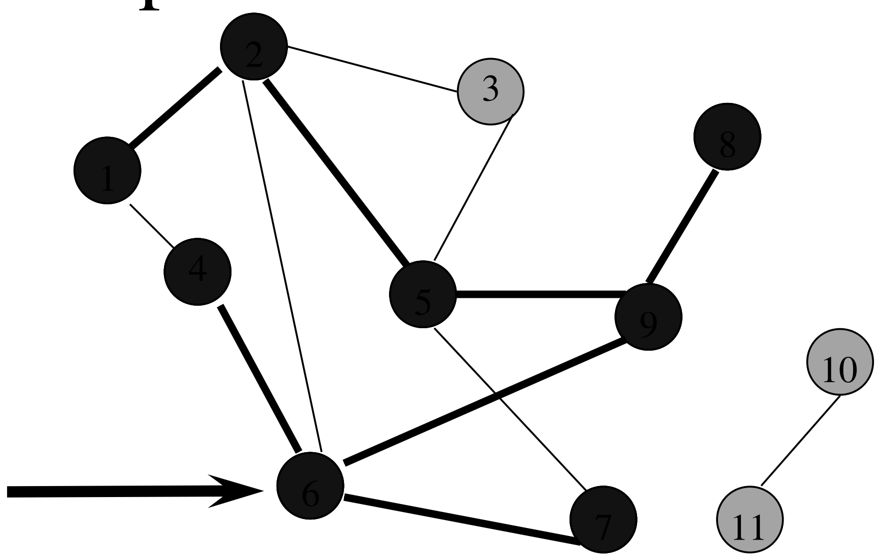
Depth-First Search Example



Label vertex 4 and return to 6.

From vertex 6 do a DFS(7).

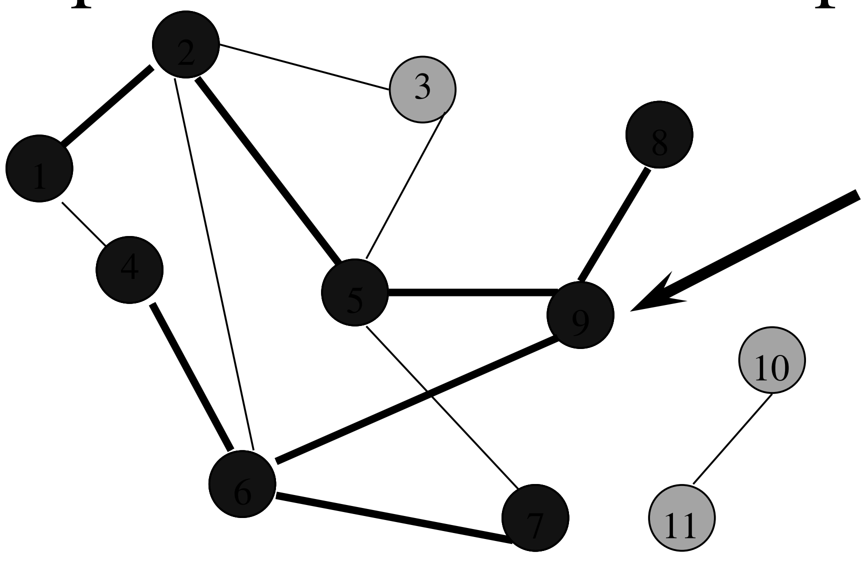
Depth-First Search Example



Label vertex 7 and return to 6.

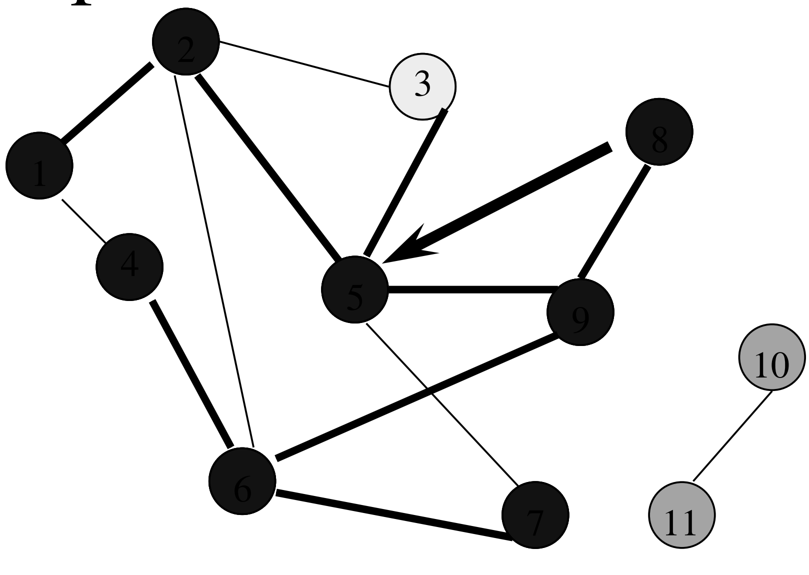
Return to 9.

Depth-First Search Example



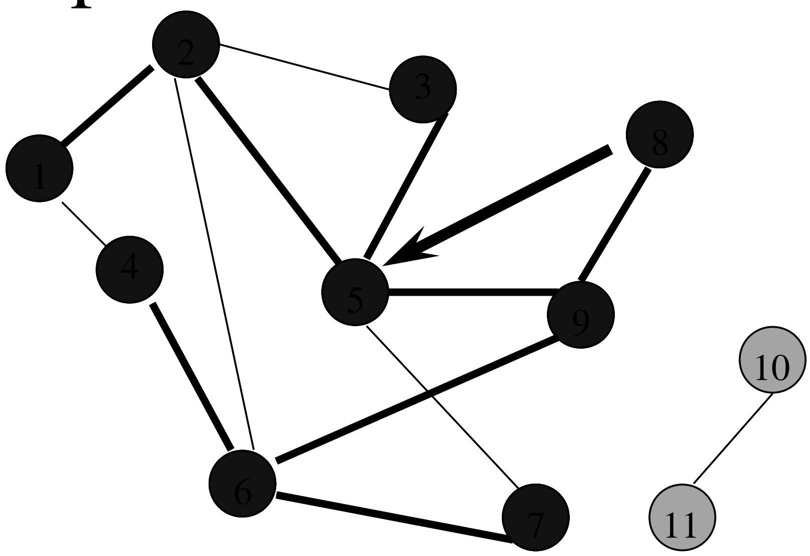
Return to 5.

Depth-First Search Example



Do a DFS(3).

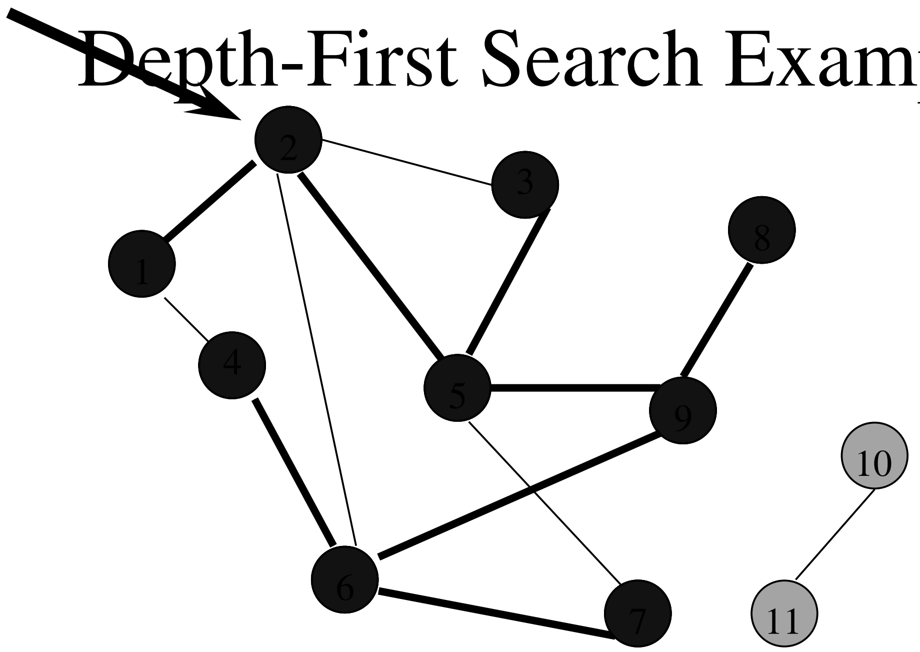
Depth-First Search Example



Label 3 and return to 5.

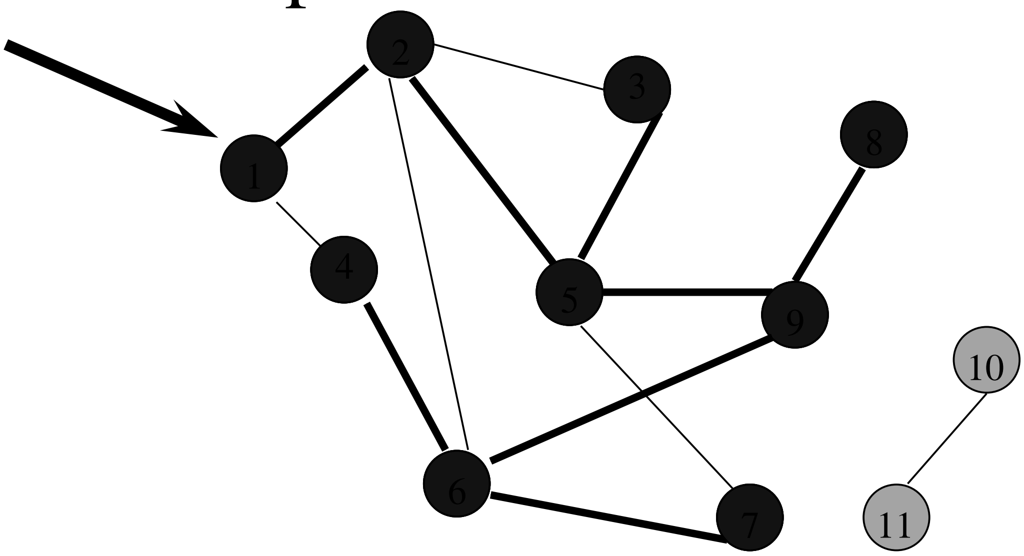
Return to 2.

Depth-First Search Example



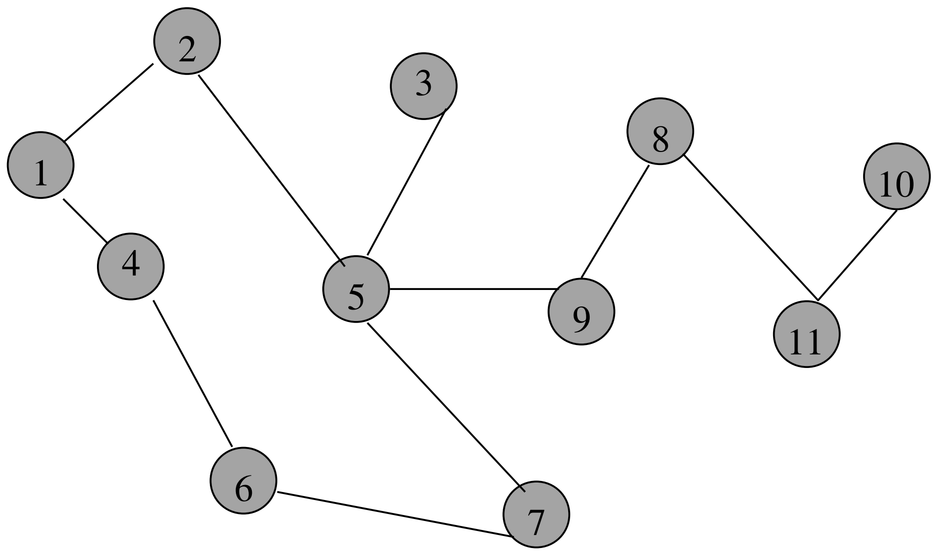
Return to 1.

Depth-First Search Example



Return to invoking method.

In Class Exercise
Do Depth first search on the following tree



Depth-First Search Property

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

Path From Vertex v To Vertex u

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

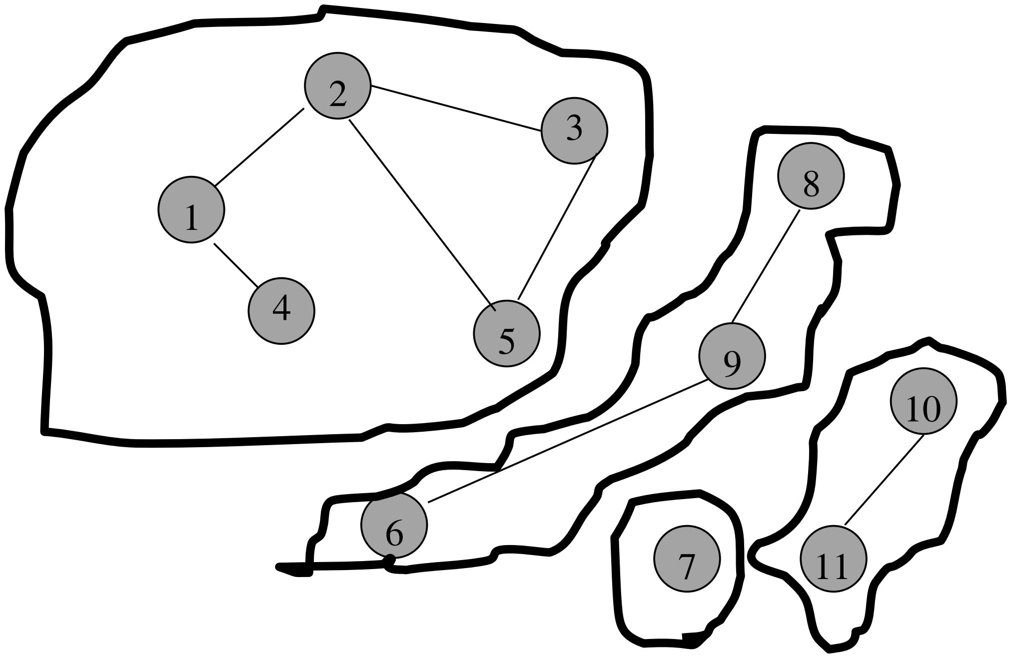
Is The Graph Connected?

- Start a depth-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 depth-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.

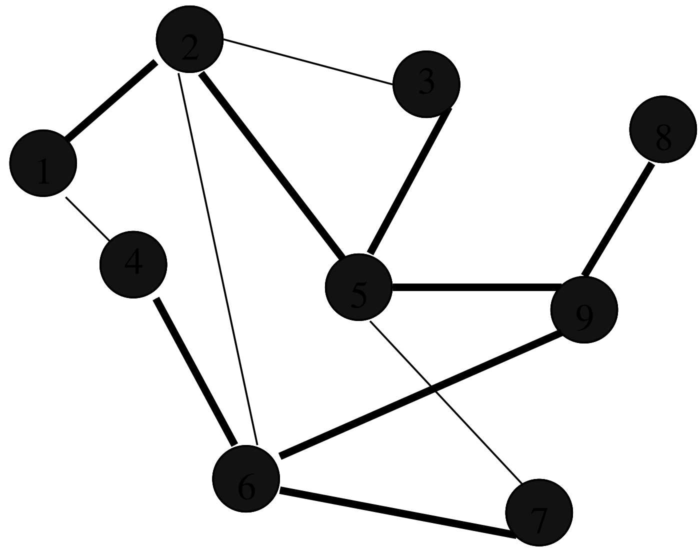
Connected Components



Time Complexity

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

Spanning Tree



Depth-first search from vertex 1.

Depth-first spanning tree.

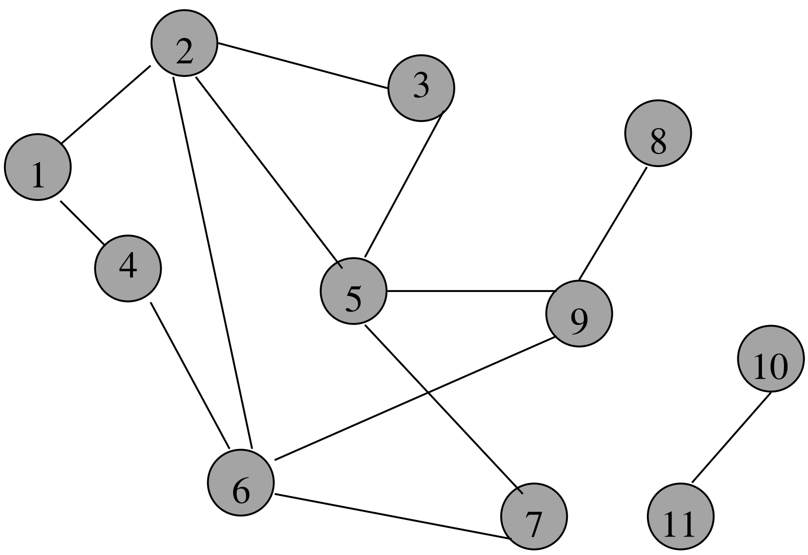
Spanning Tree

- Start a depth-first search at any vertex of the graph.
- If graph is connected, the n-1 edges used to get to unvisited vertices define a spanning tree (depth-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

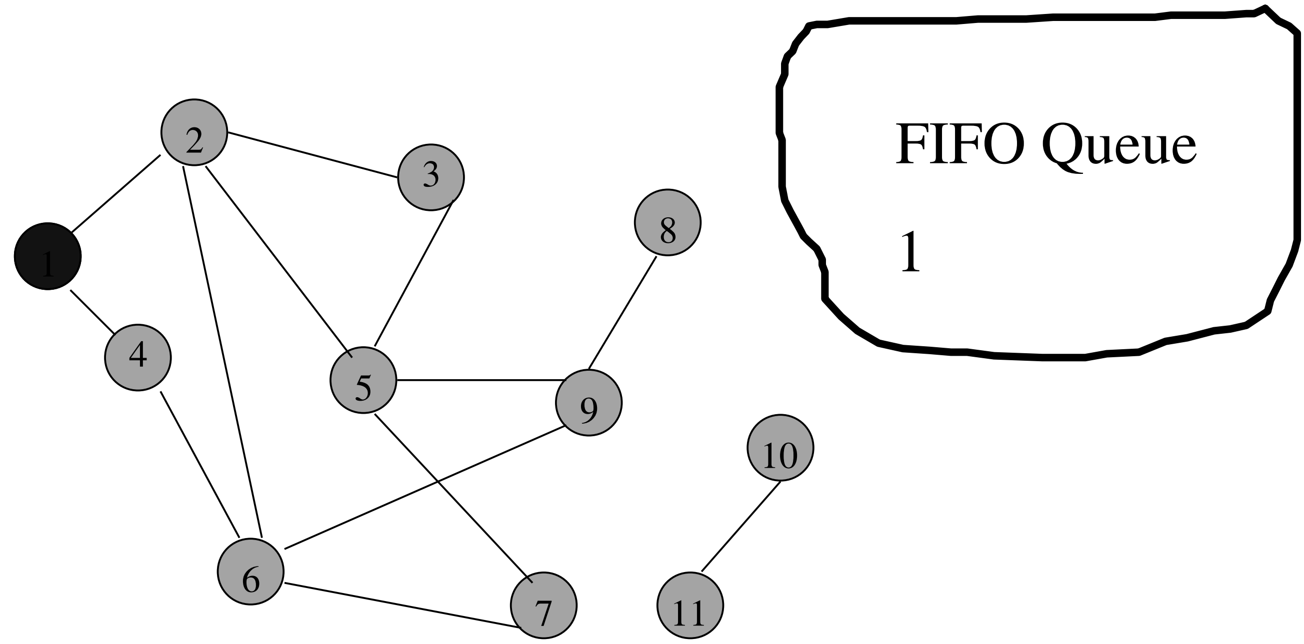
- Visit start vertex and put into a FIFO queue.
- Repeatedly remove a vertex from the queue, visit its unvisited adjacent vertices, put newly visited vertices into the queue.

Breadth-First Search Example



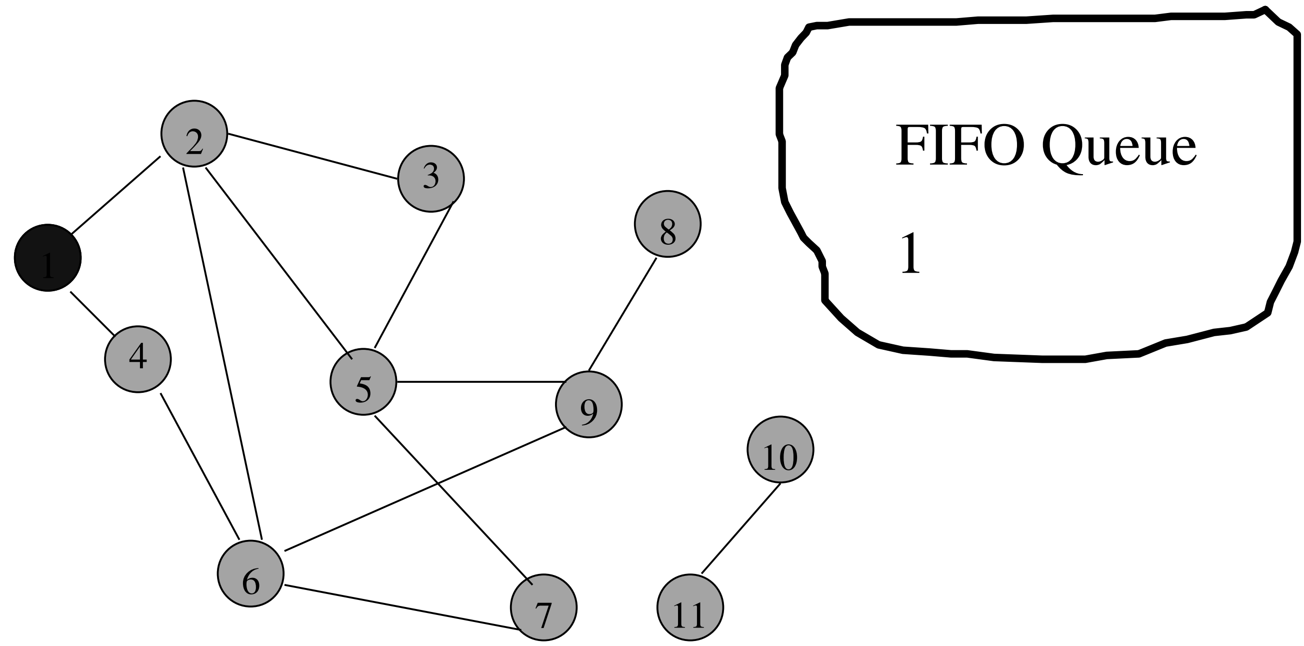
Start search at vertex 1.

Breadth-First Search Example



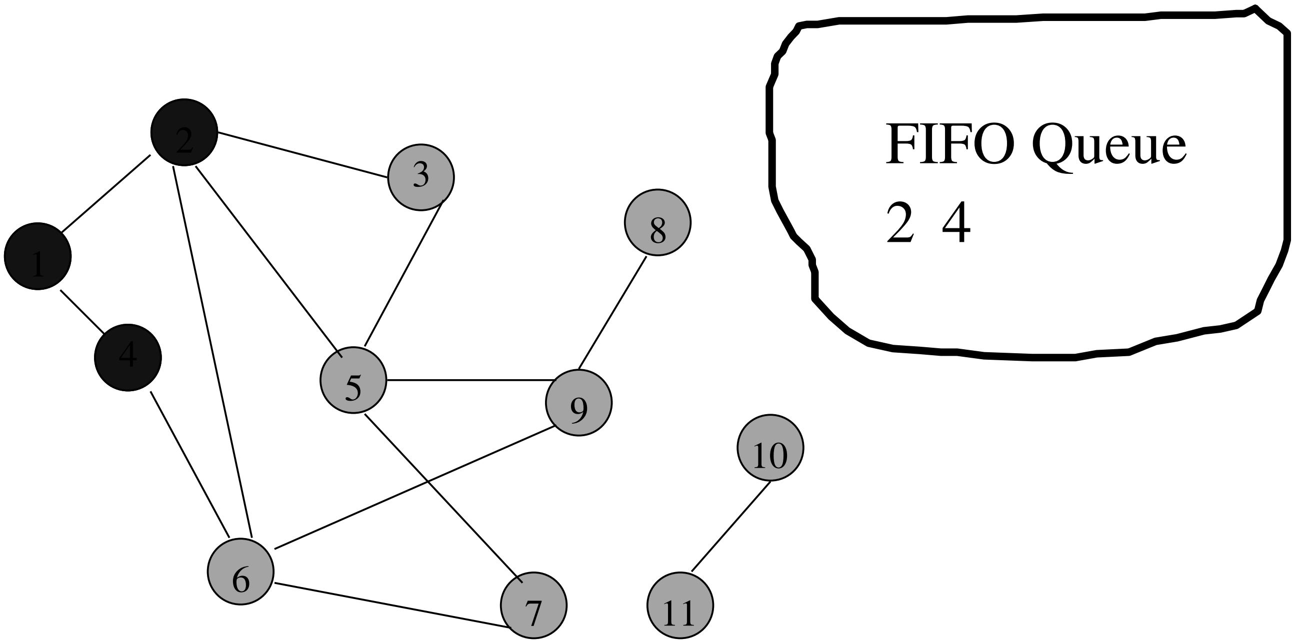
Visit/mark/label start vertex and put in a FIFO queue.

Breadth-First Search Example



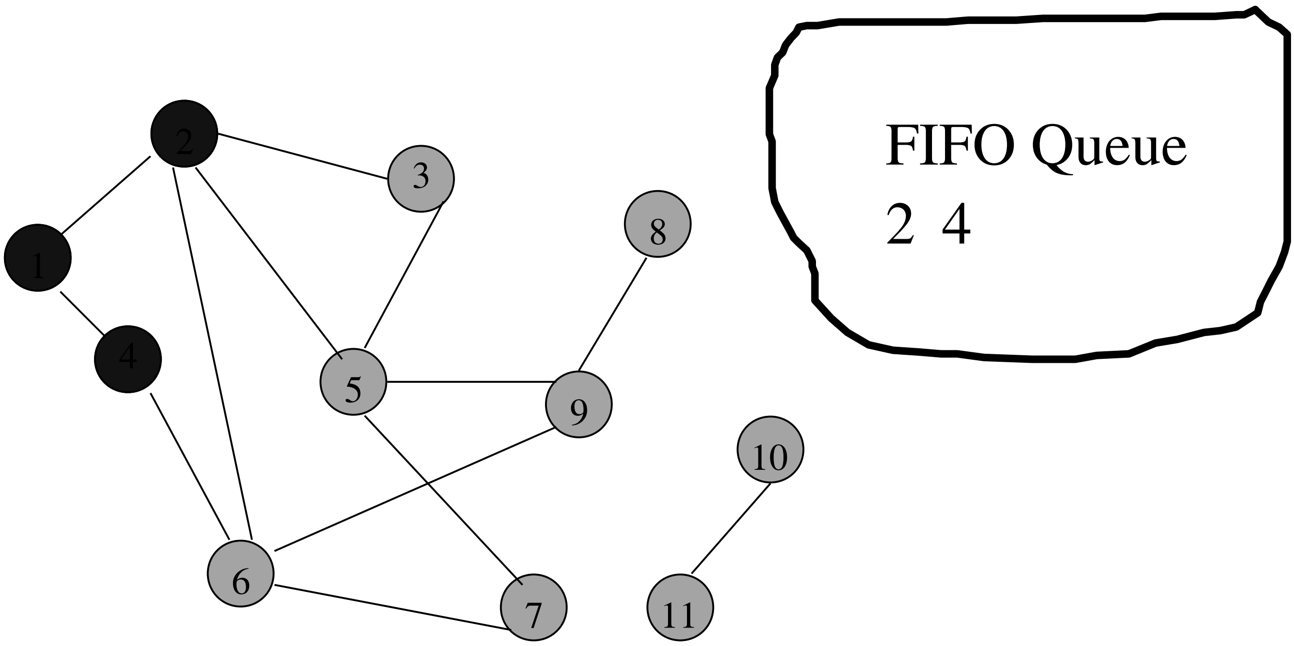
Remove 1 from Q; visit adjacent unvisited vertices;
put in Q.

Breadth-First Search Example



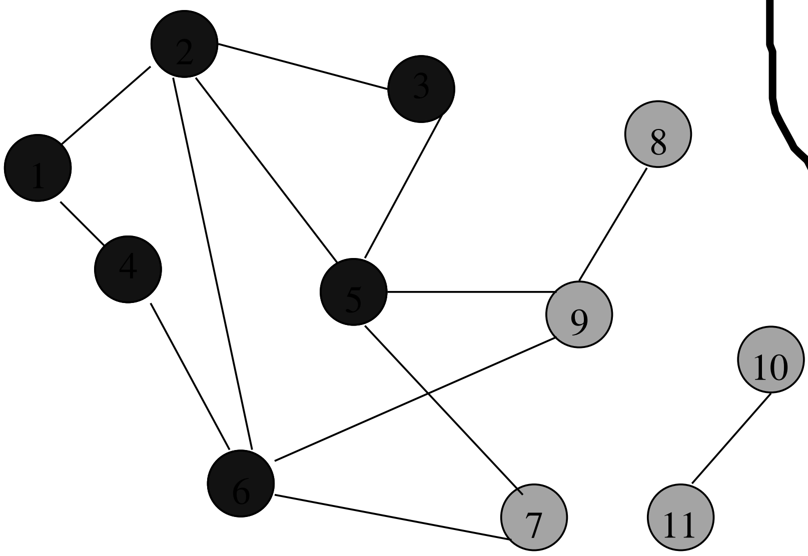
Remove 1 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example



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

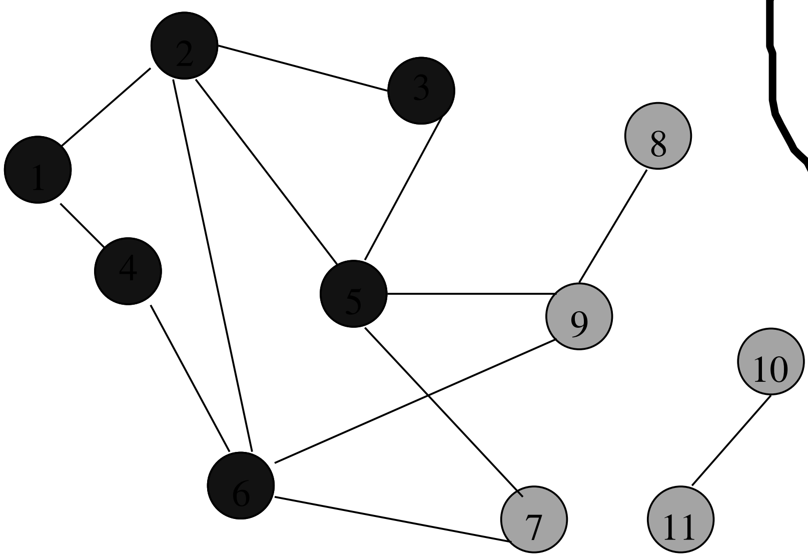
Breadth-First Search Example



FIFO Queue
4 5 3 6

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

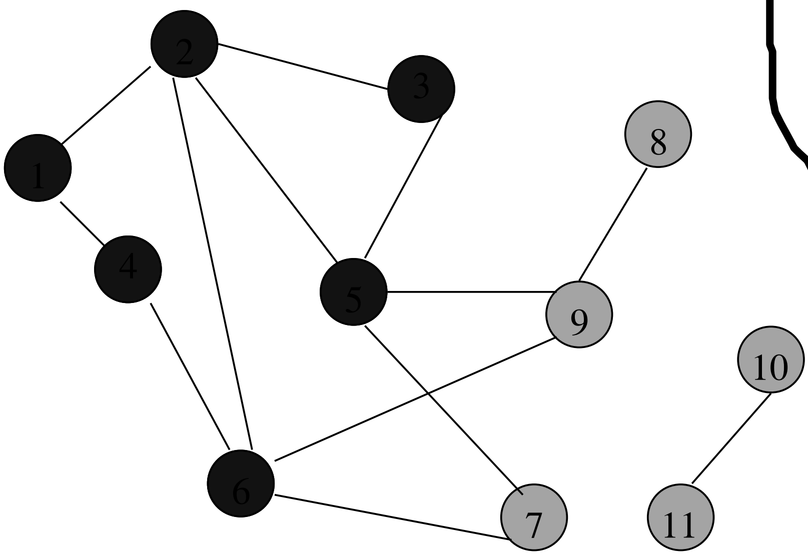
Breadth-First Search Example



FIFO Queue
4 5 3 6

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

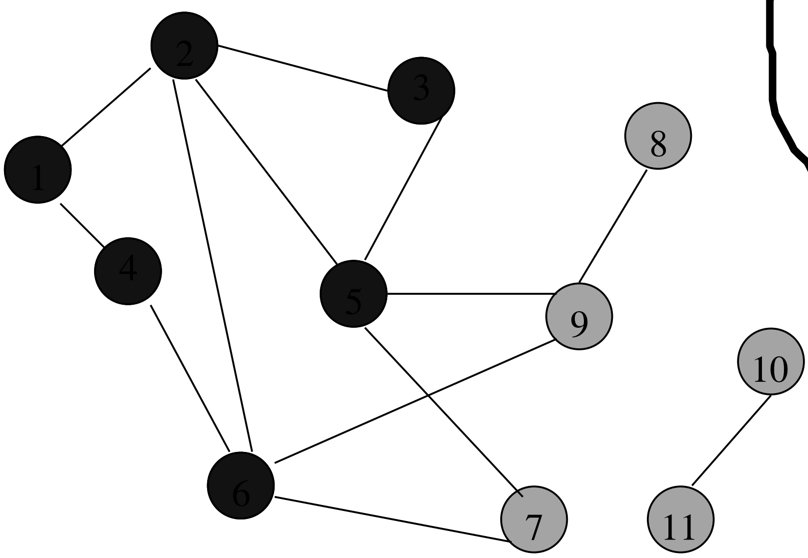
Breadth-First Search Example



FIFO Queue
5 3 6

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

Breadth-First Search Example



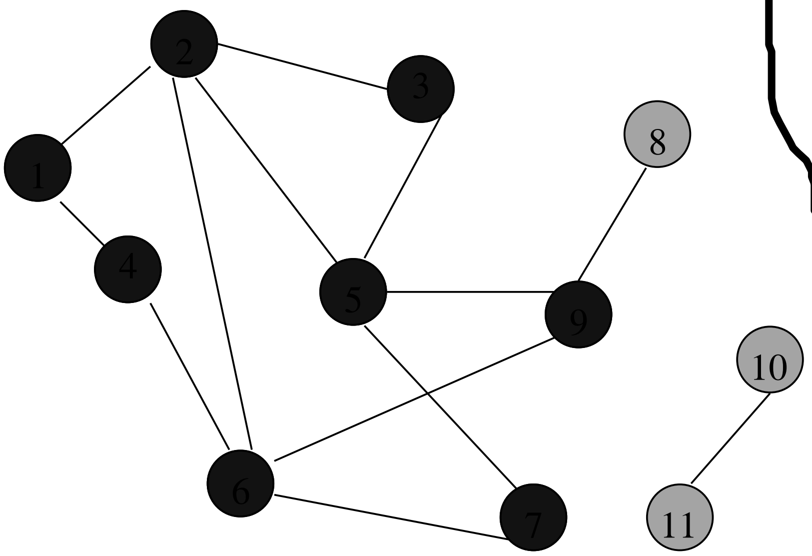
FIFO Queue
5 3 6

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

Breadth-First Search Example

FIFO Queue

3 6 9 7

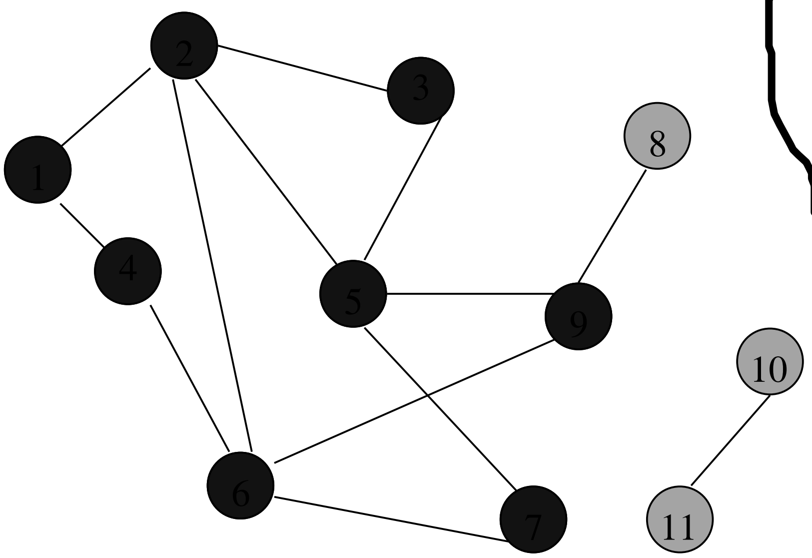


Remove 5 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example

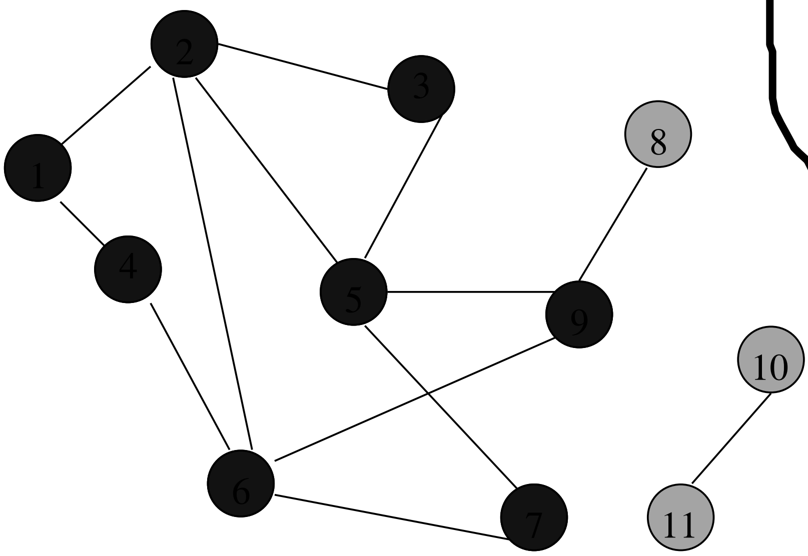
FIFO Queue

3 6 9 7



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

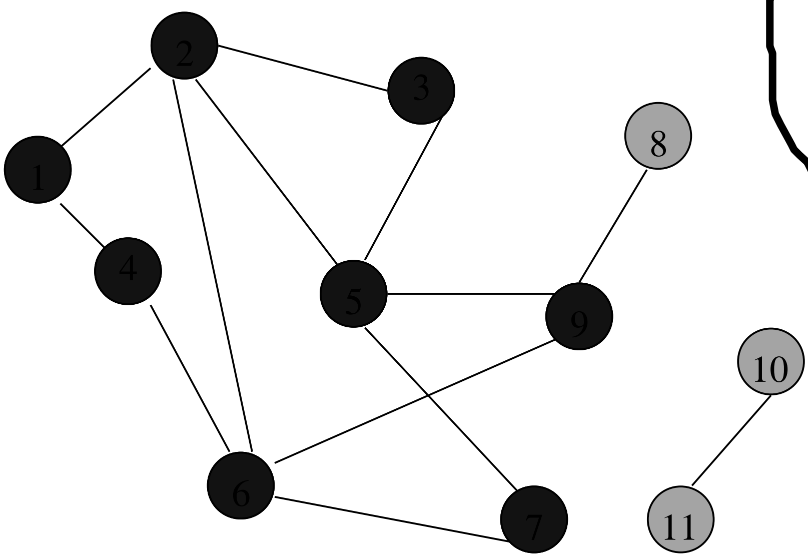
Breadth-First Search Example



FIFO Queue
6 9 7

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

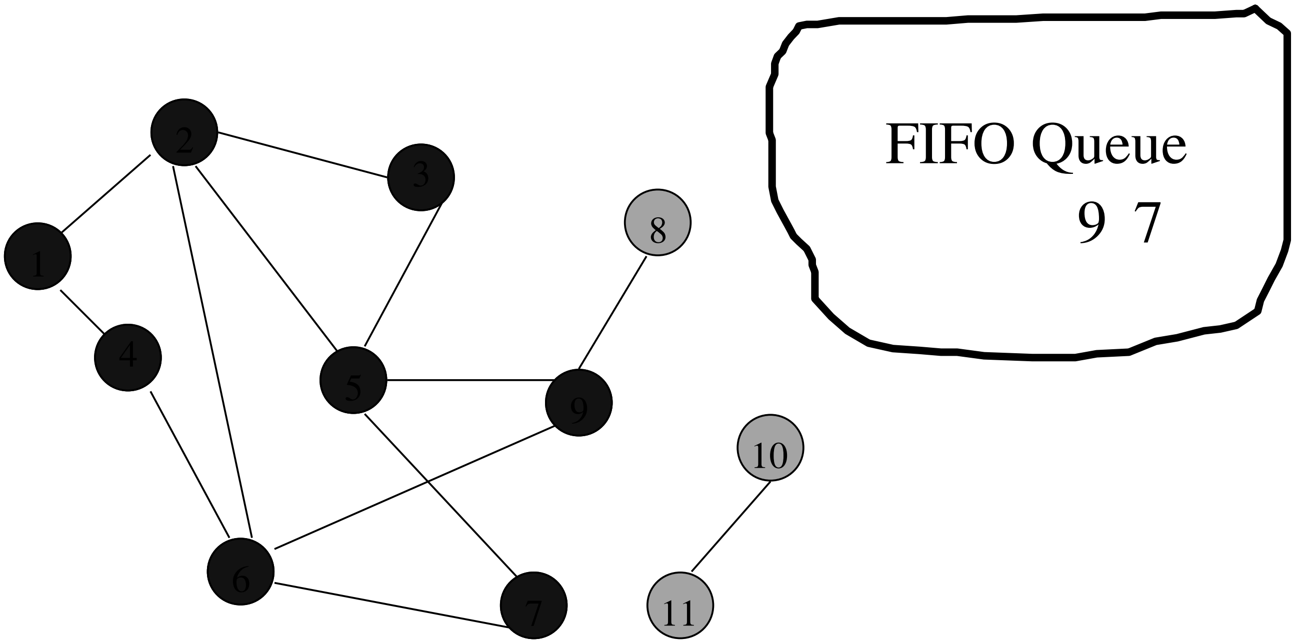
Breadth-First Search Example



FIFO Queue
6 9 7

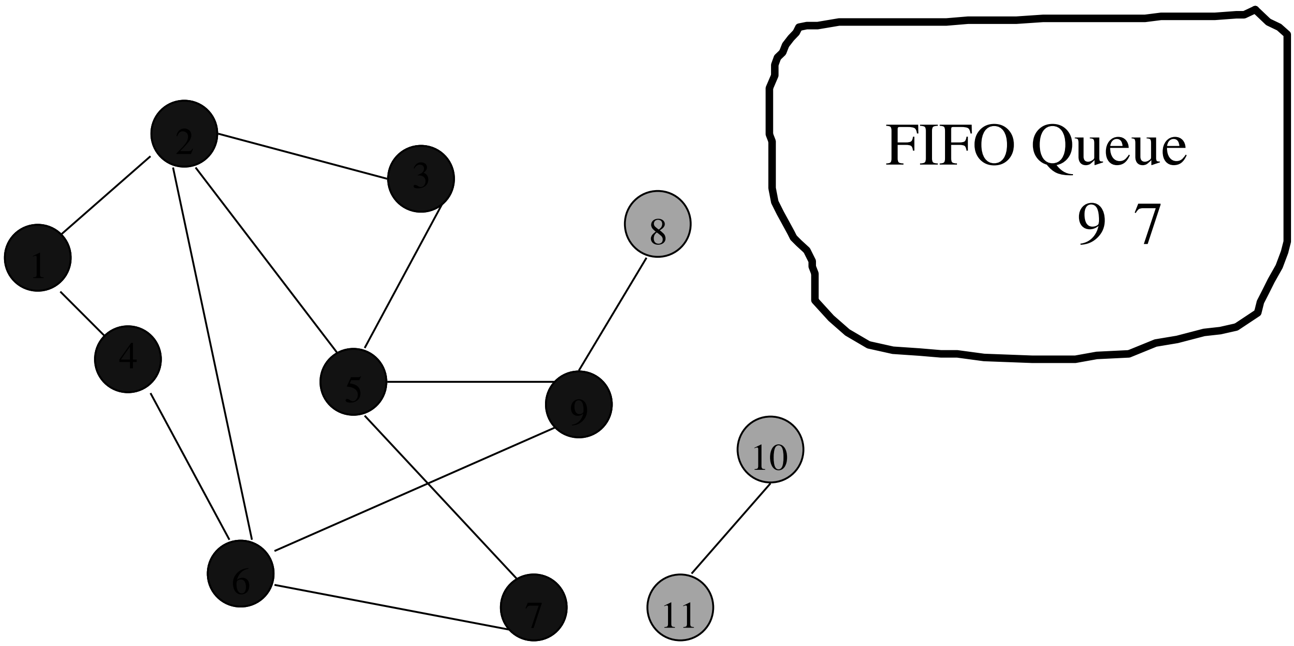
Remove 6 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example



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

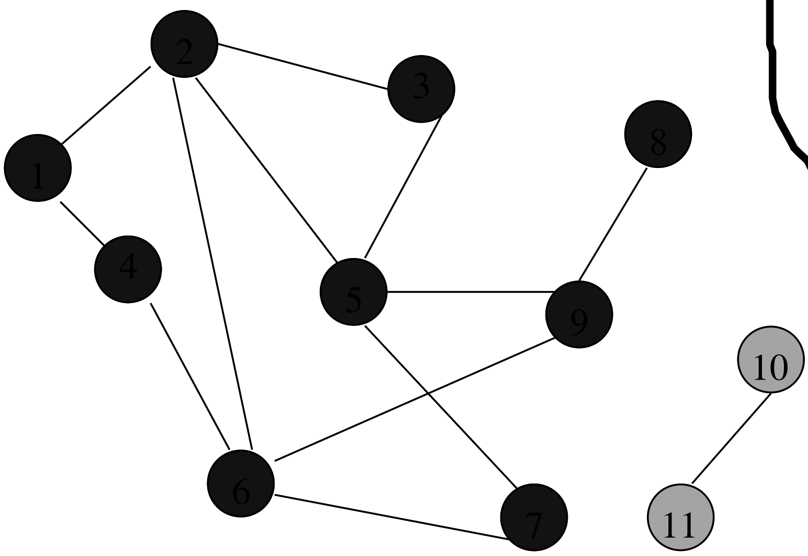
Breadth-First Search Example



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

Breadth-First Search Example

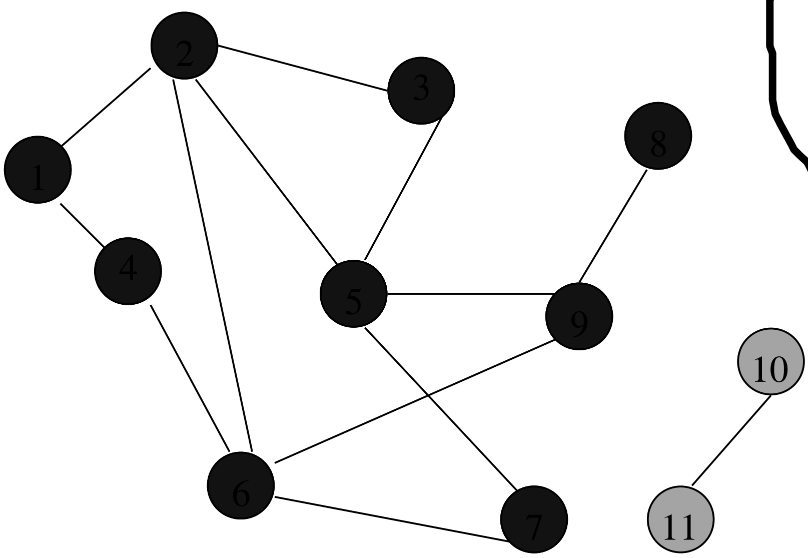
FIFO Queue
7 8



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

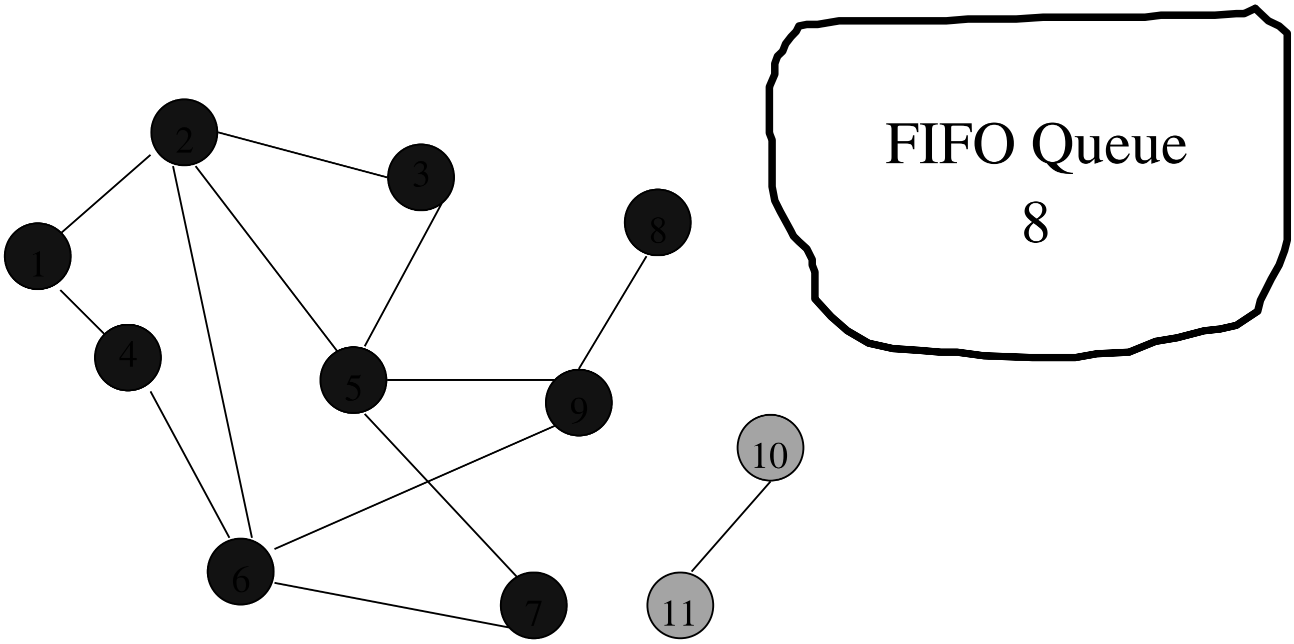
Breadth-First Search Example

FIFO Queue
7 8



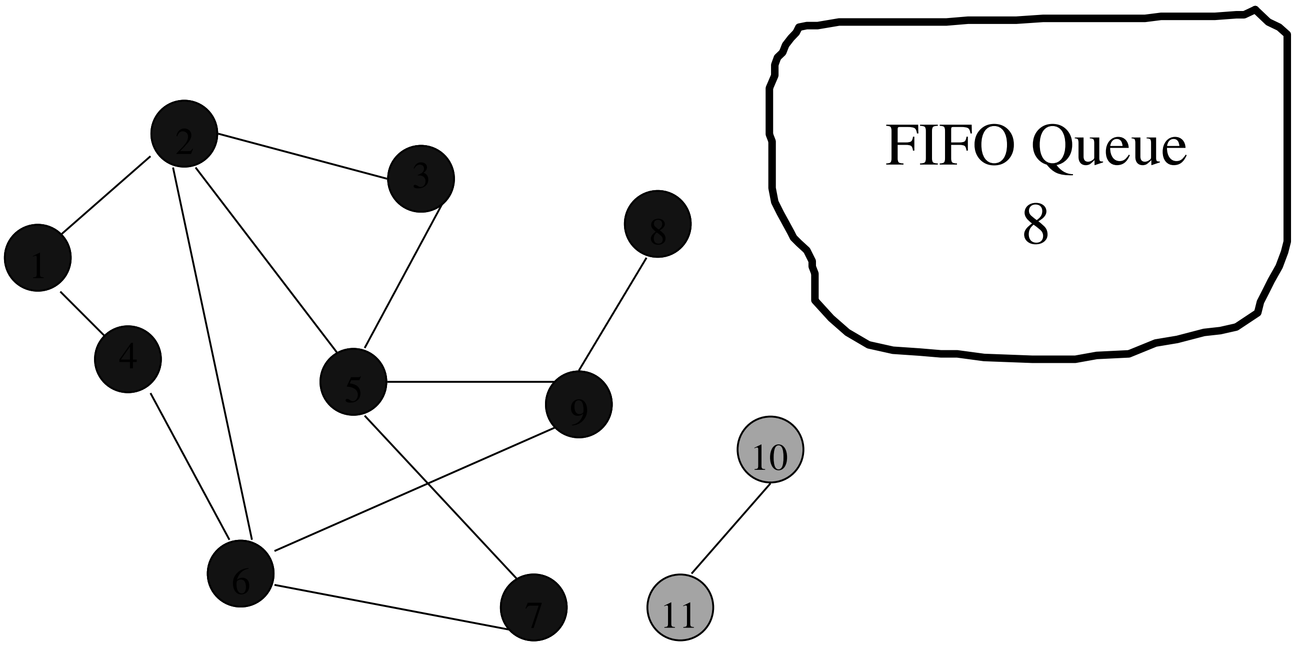
Remove 7 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example



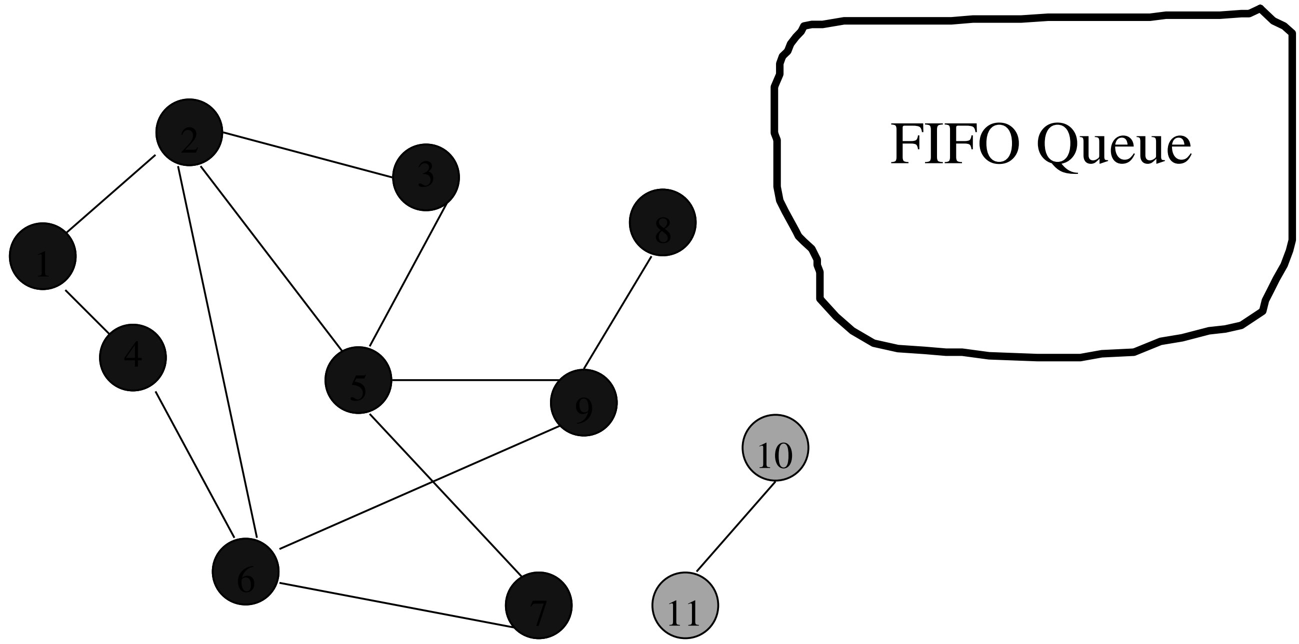
Remove 7 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example



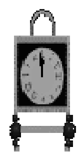
Remove 8 from Q; visit adjacent unvisited vertices; put in Q.

Breadth-First Search Example



Queue is empty. Search terminates.

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
  - 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)

Breadth-First Search Properties

- Same complexity as DFS.
- Same properties with respect to path finding, connected components, and spanning trees.
- There are problems for which bfs is better than dfs and vice versa.

Homework

- Sec. 6.2 Exercise2 @P352