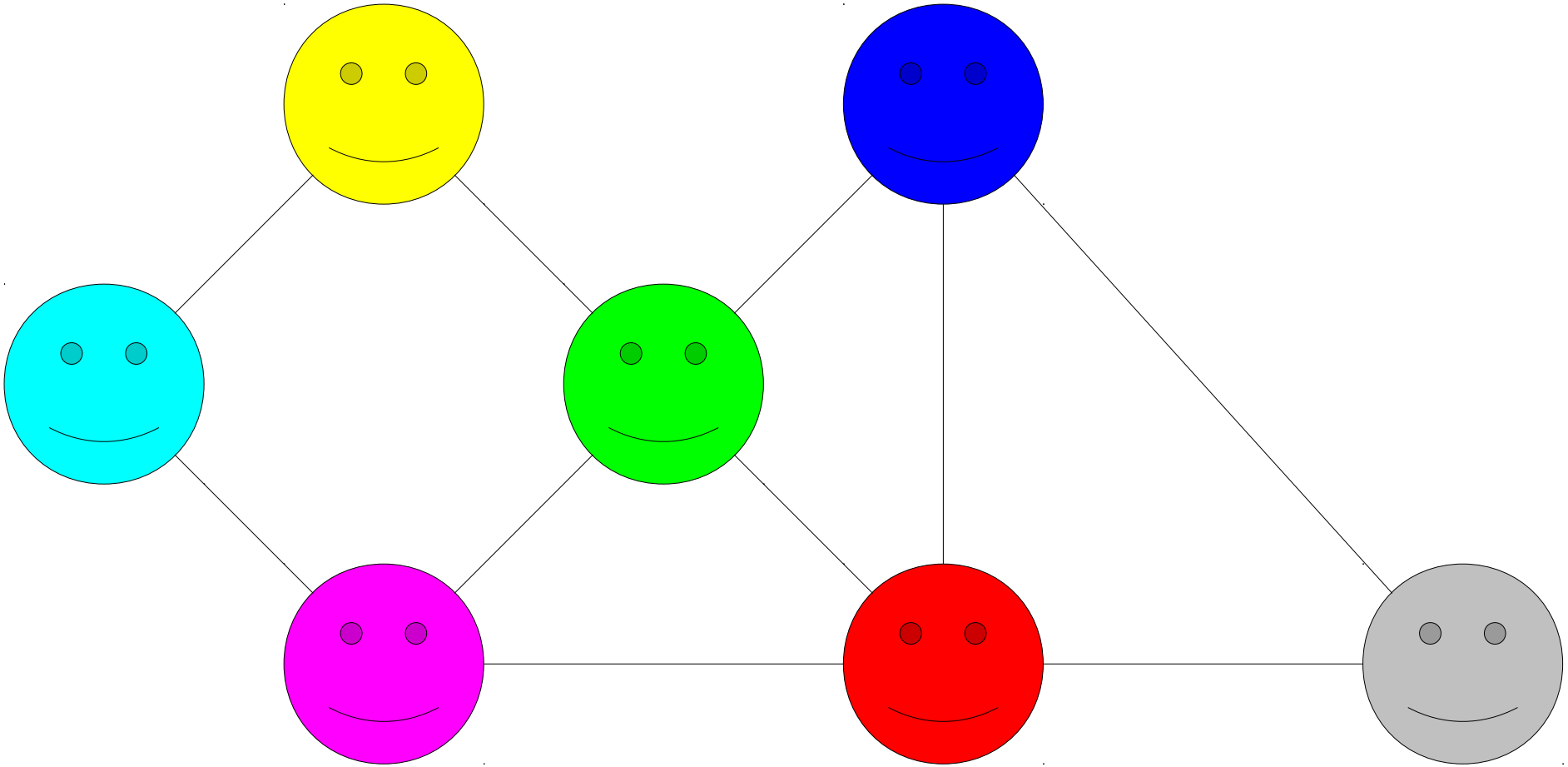
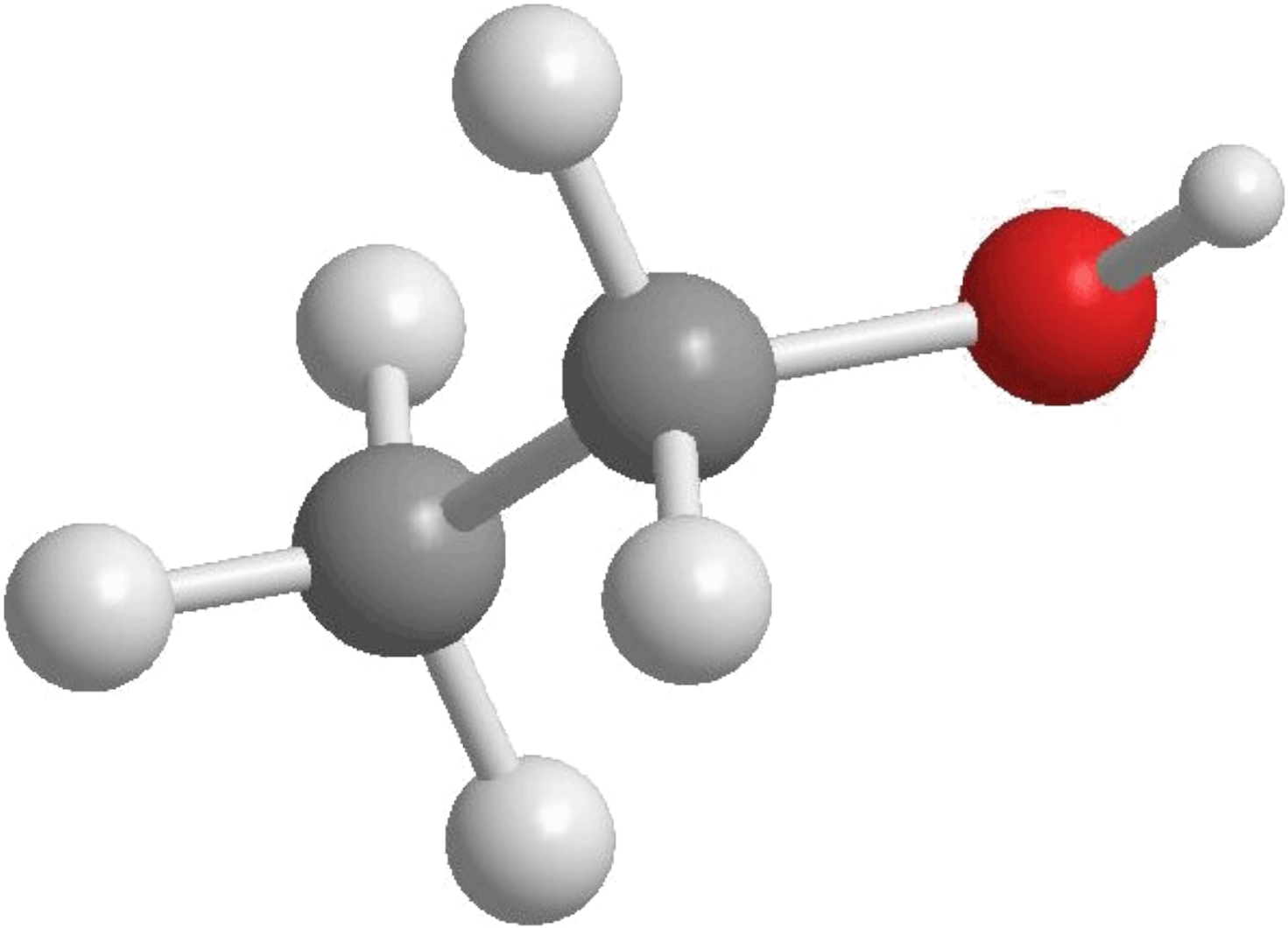
Graphs

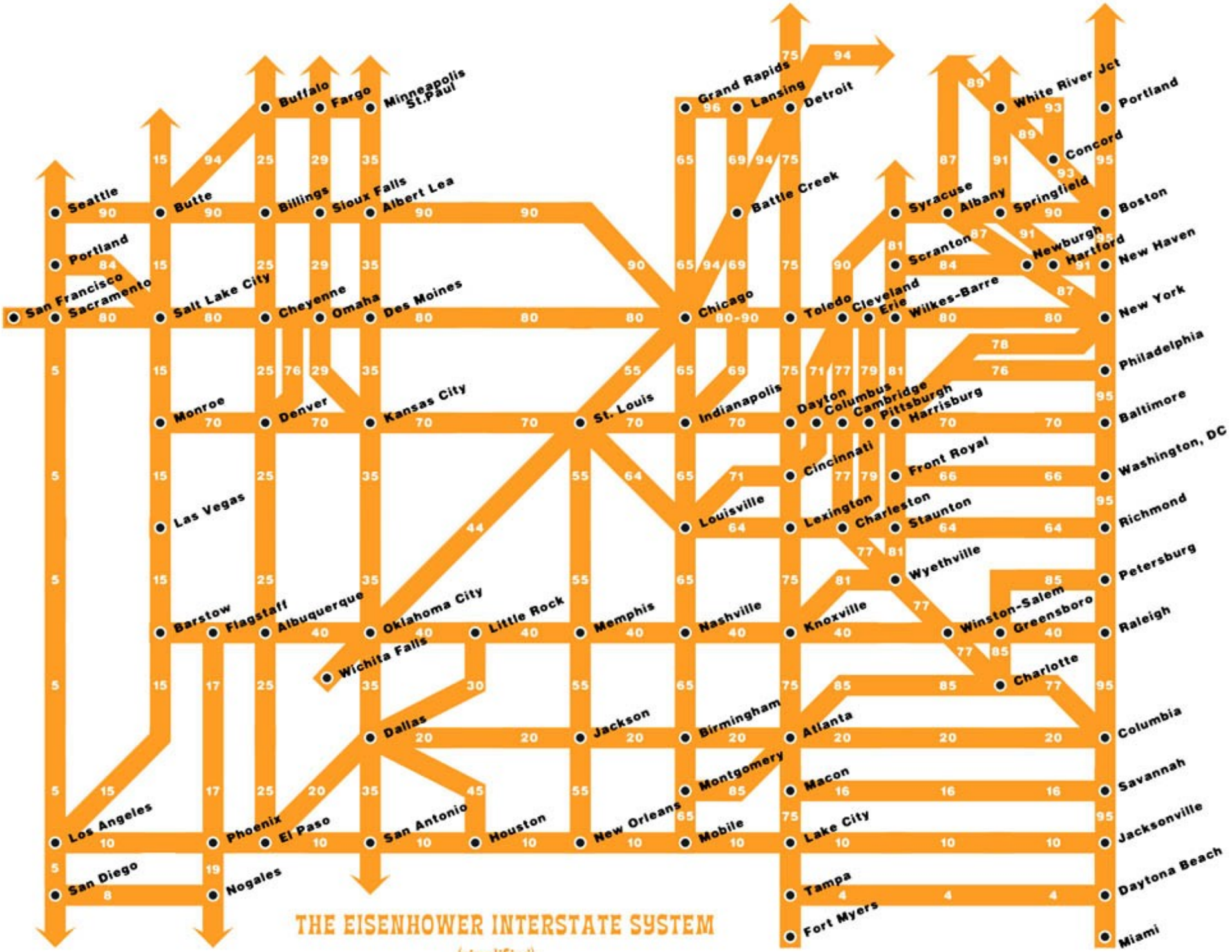
A Social Network



Chemical Bonds

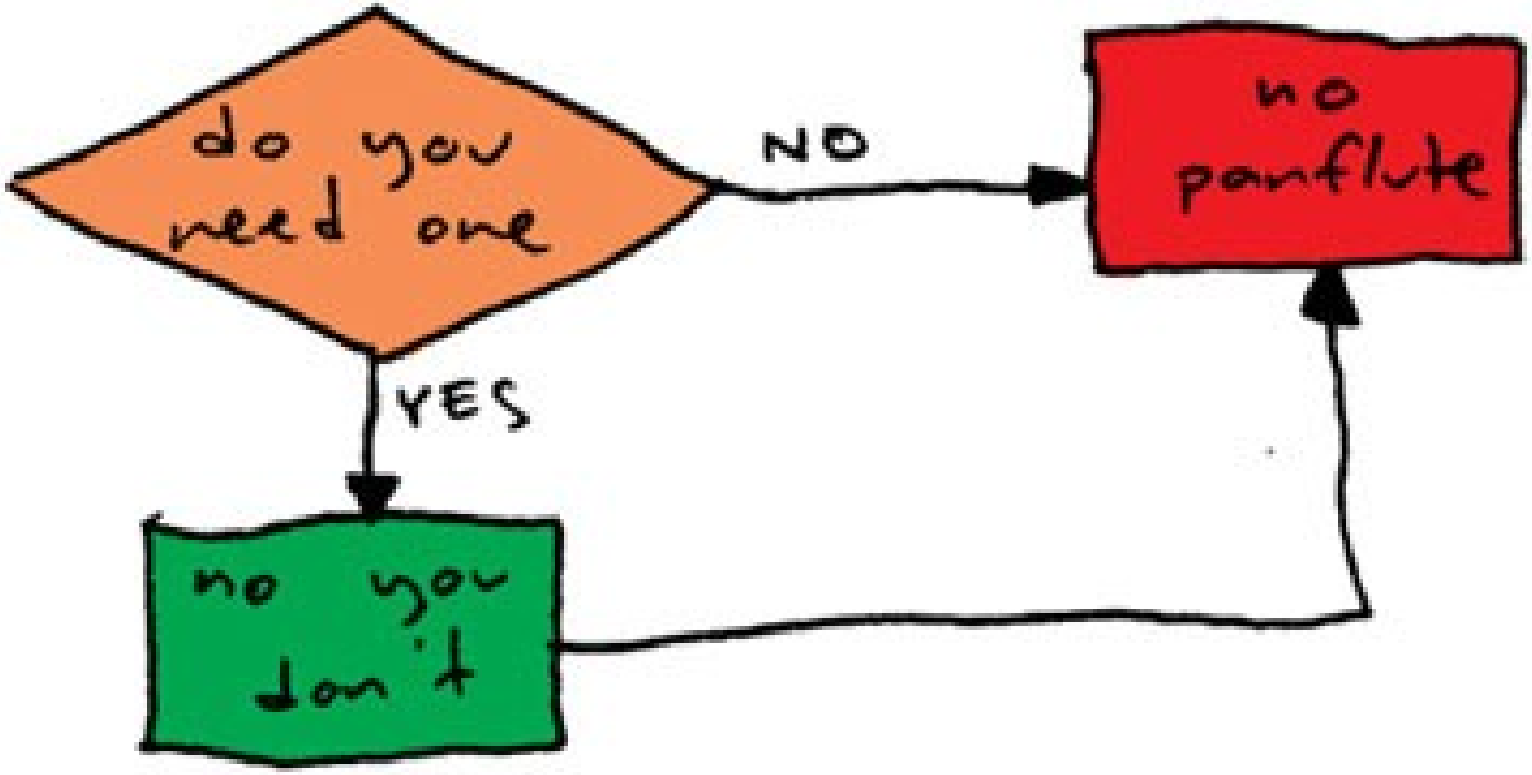


http://4.bp.blogspot.com/-xCtBJ8lKHqA/Tjm0BONWBRI/AAAAAAAAAK4/-mHrbAUOHHg/s1600/Ethanol2.gif



http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg

PANFLUTE FLOWCHART

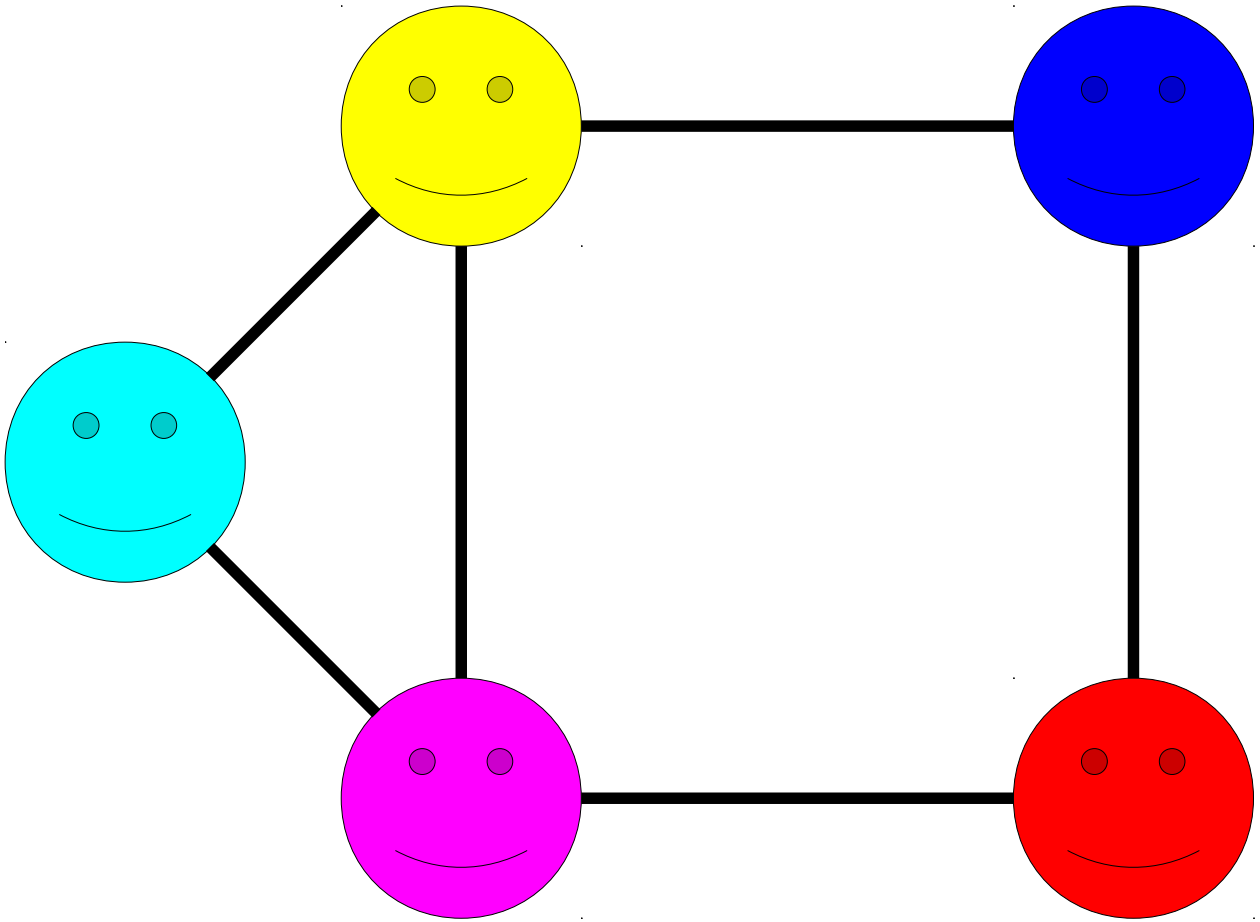


http://www.toothpastefordinner.com/

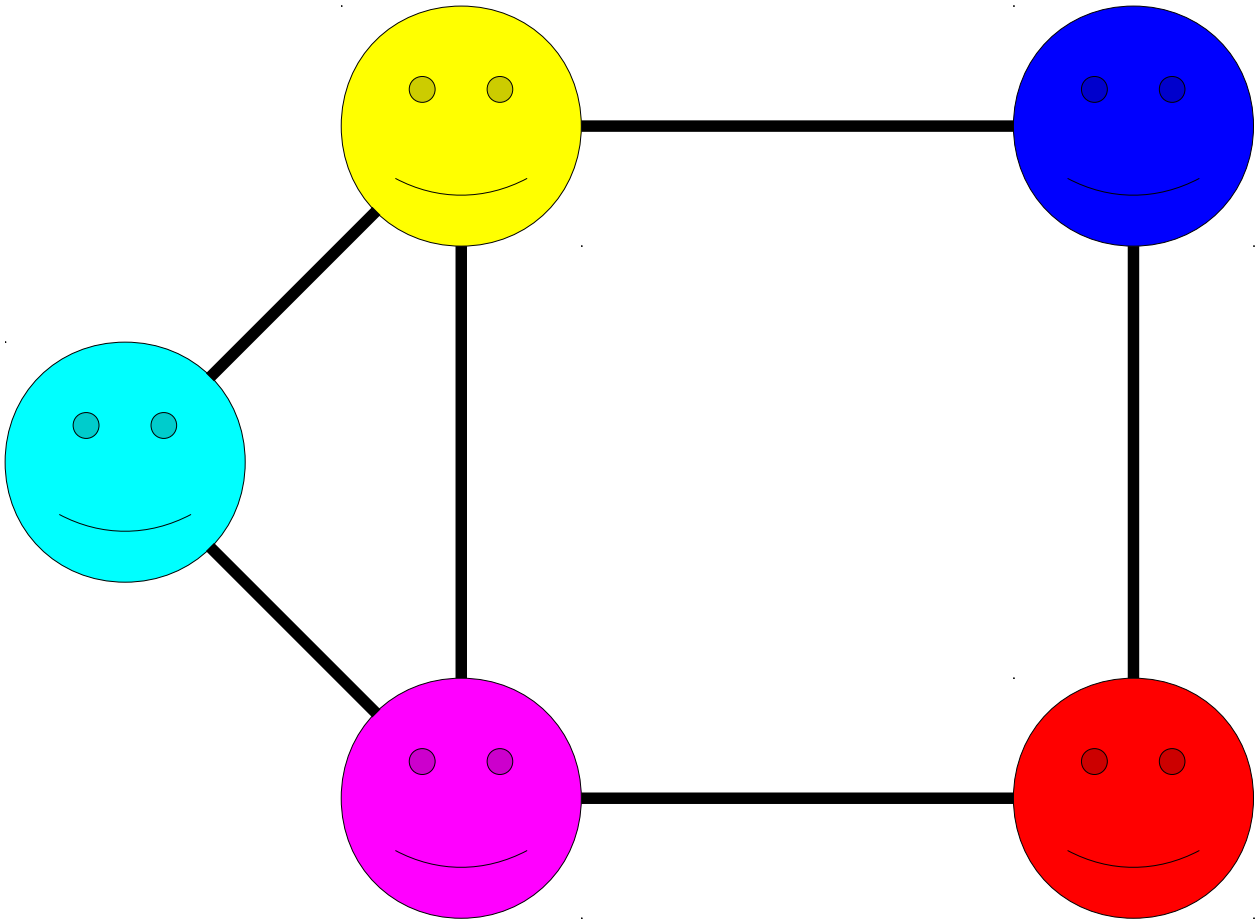
A **graph** is a mathematical structure for representing relationships.

A **graph** is a mathematical structure for representing relationships.

A **graph** is a mathematical structure for representing relationships.

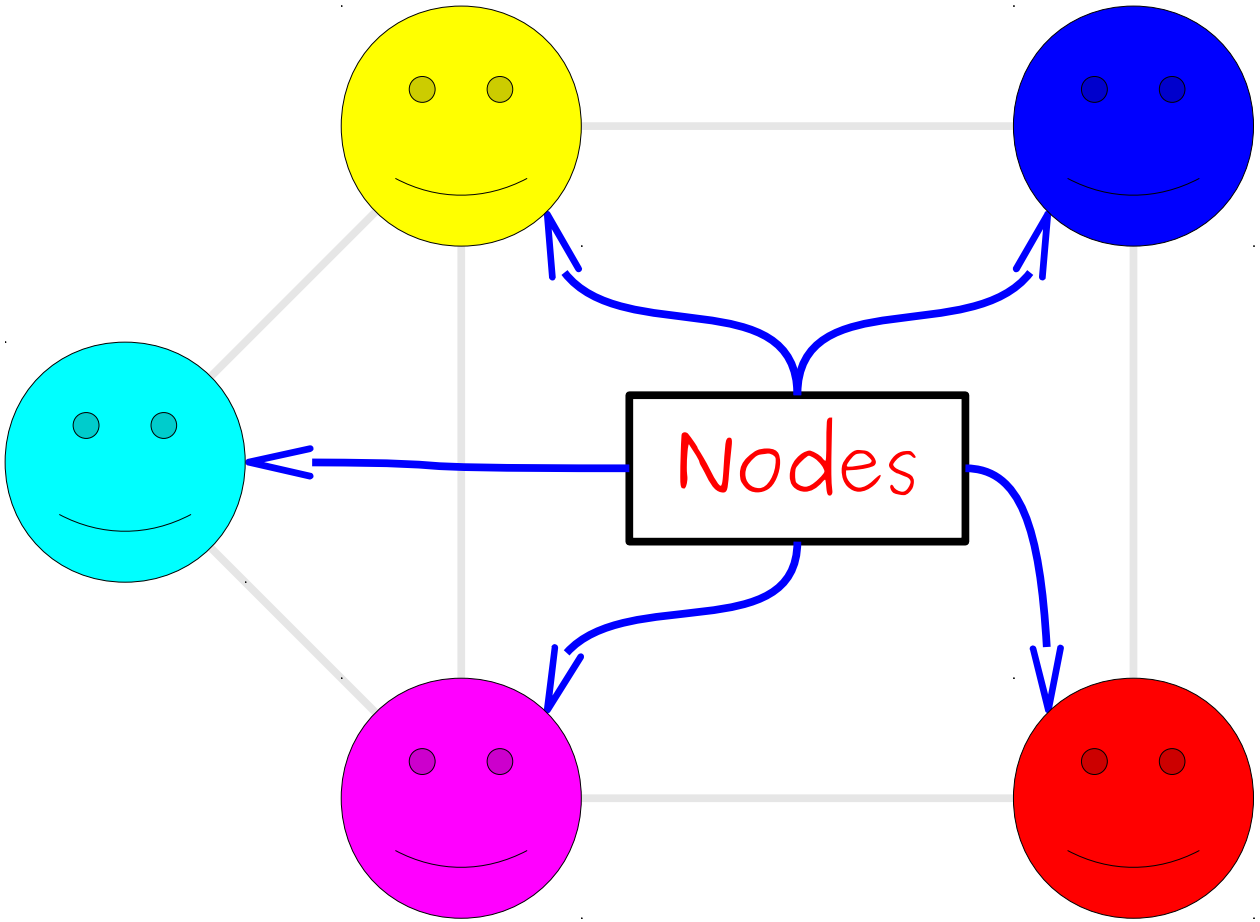


A **graph** is a mathematical structure for representing relationships.



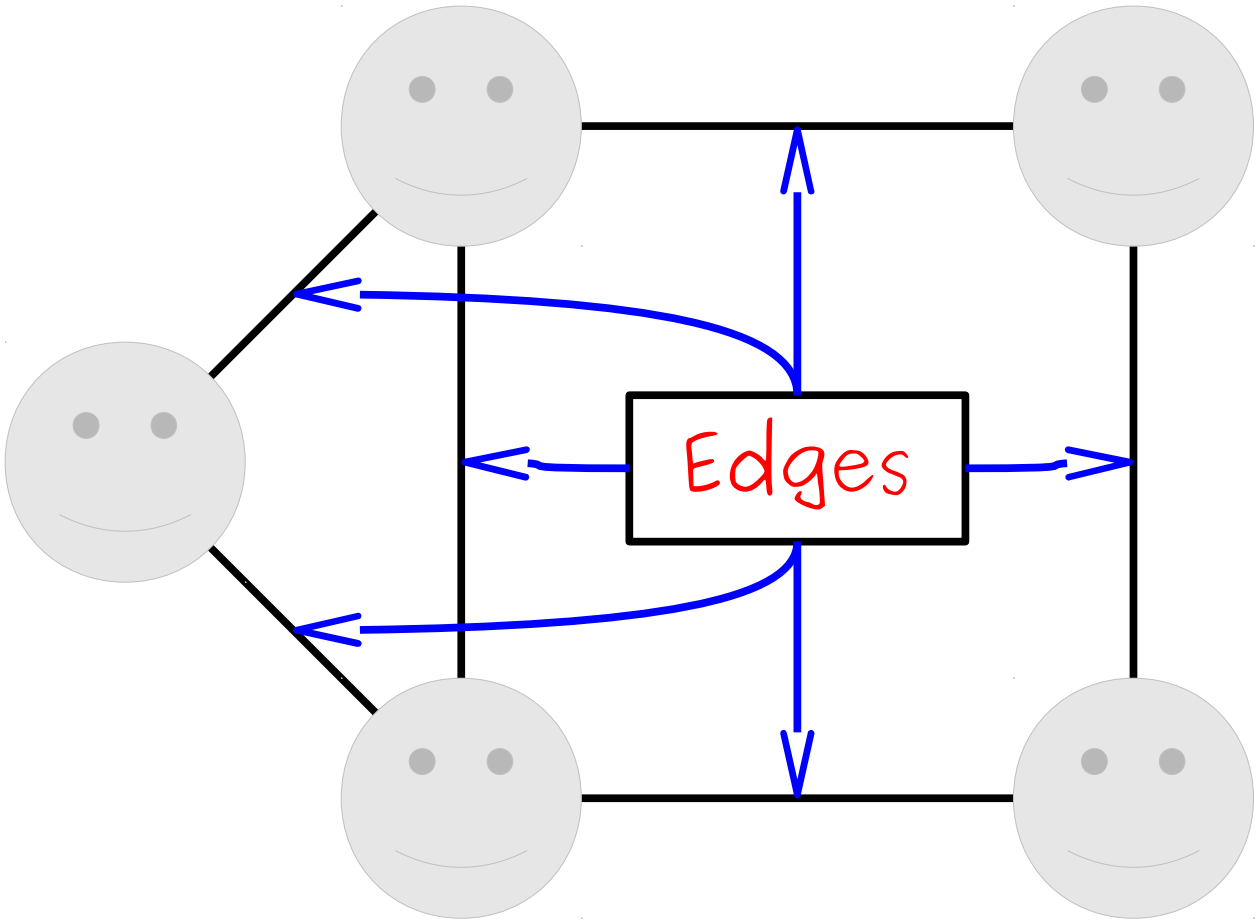
A graph consists of a set of **nodes** connected by **edges**.

A **graph** is a mathematical structure for representing relationships.



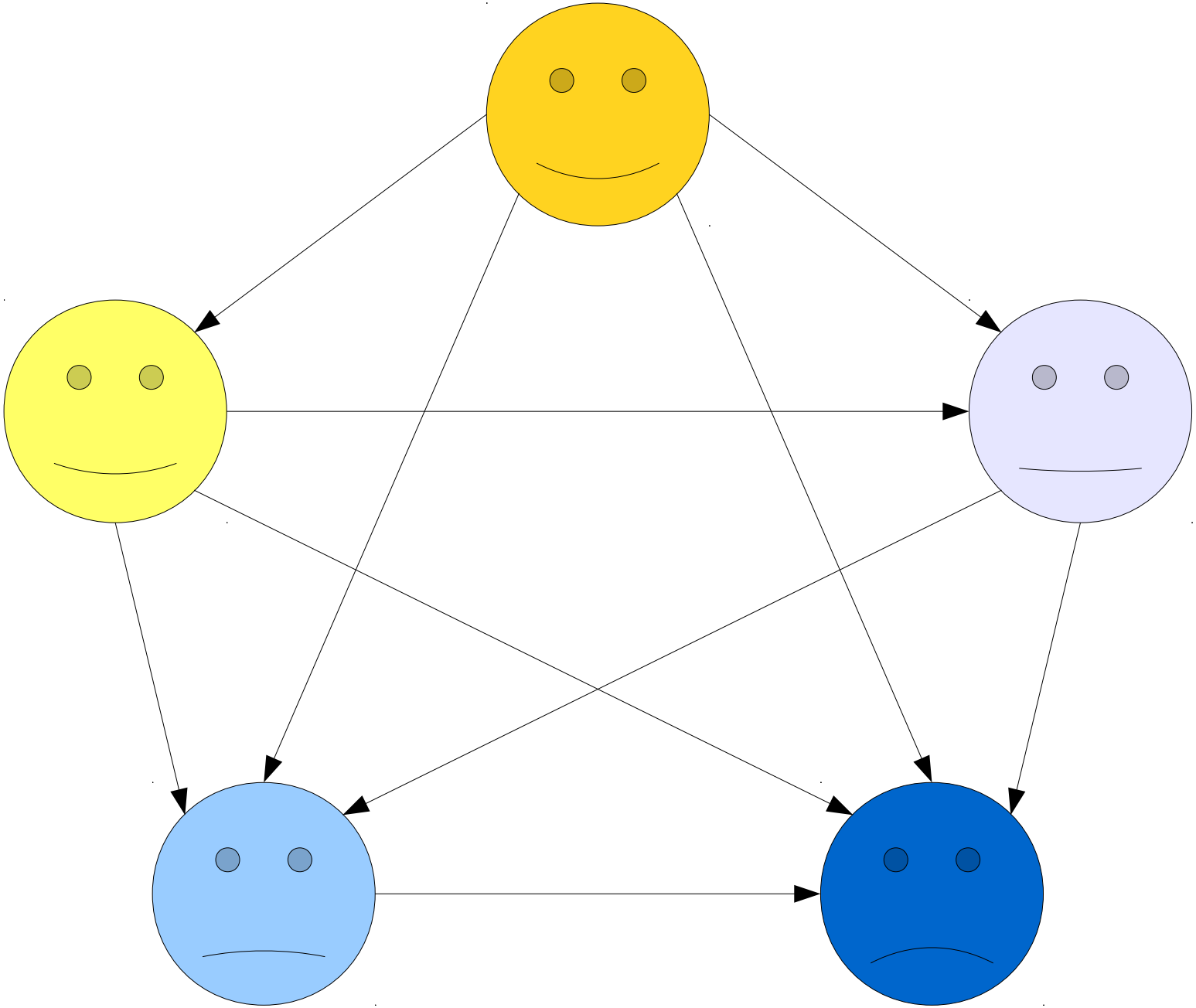
A graph consists of a set of **nodes** connected by **edges**.

A **graph** is a mathematical structure for representing relationships.

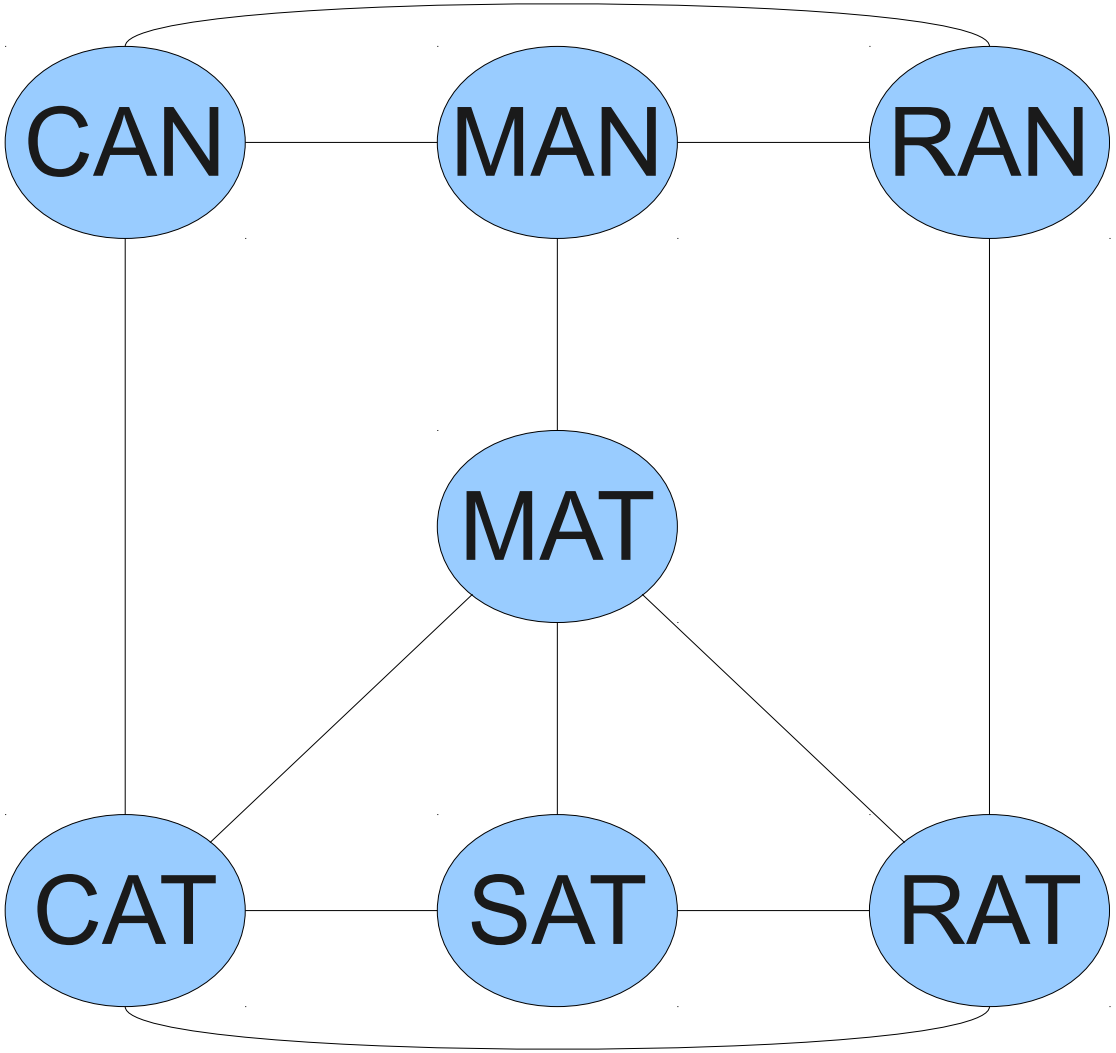


A graph consists of a set of **nodes** connected by **edges**.

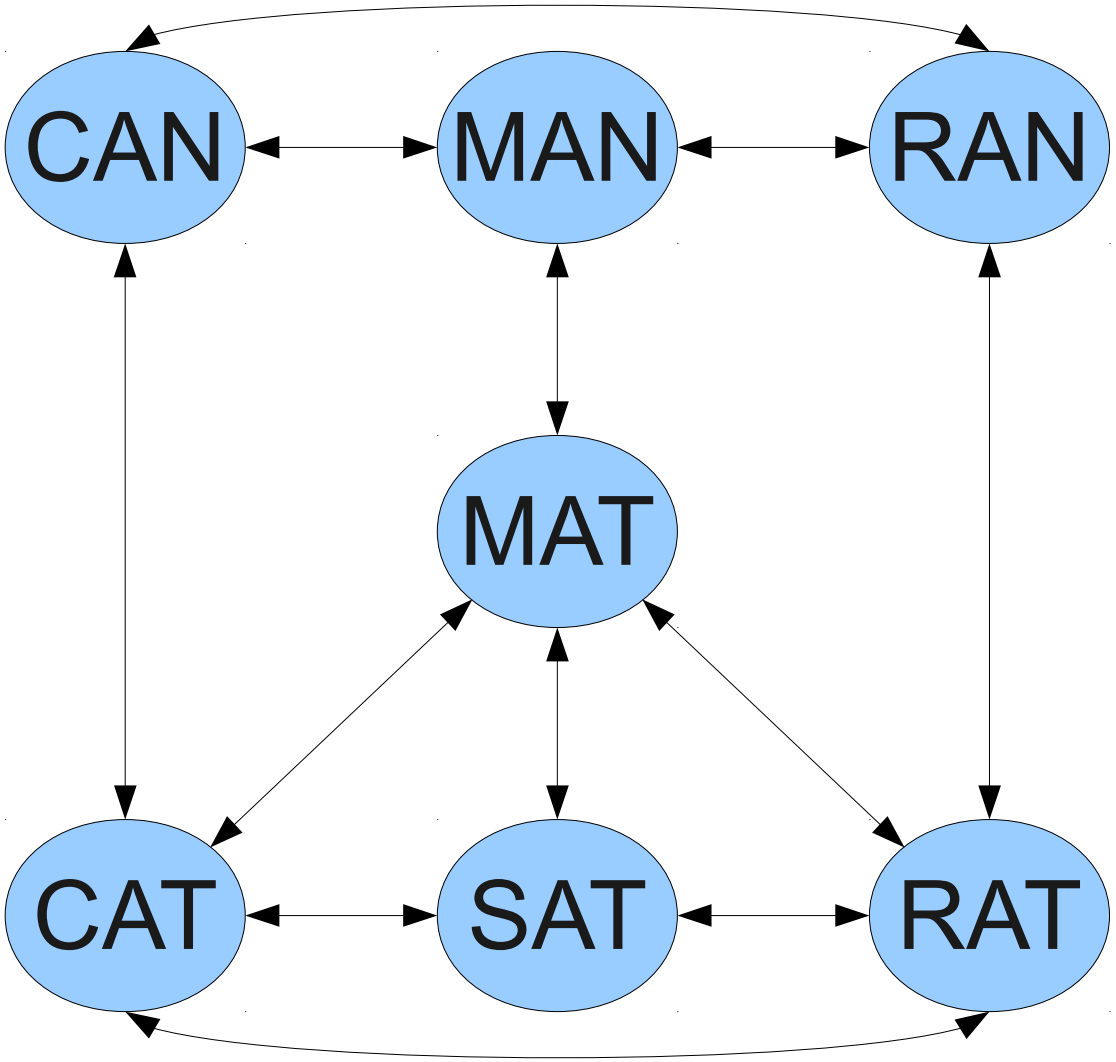
Some graphs are **directed**.



Some graphs are **undirected**.



Some graphs are **undirected**.

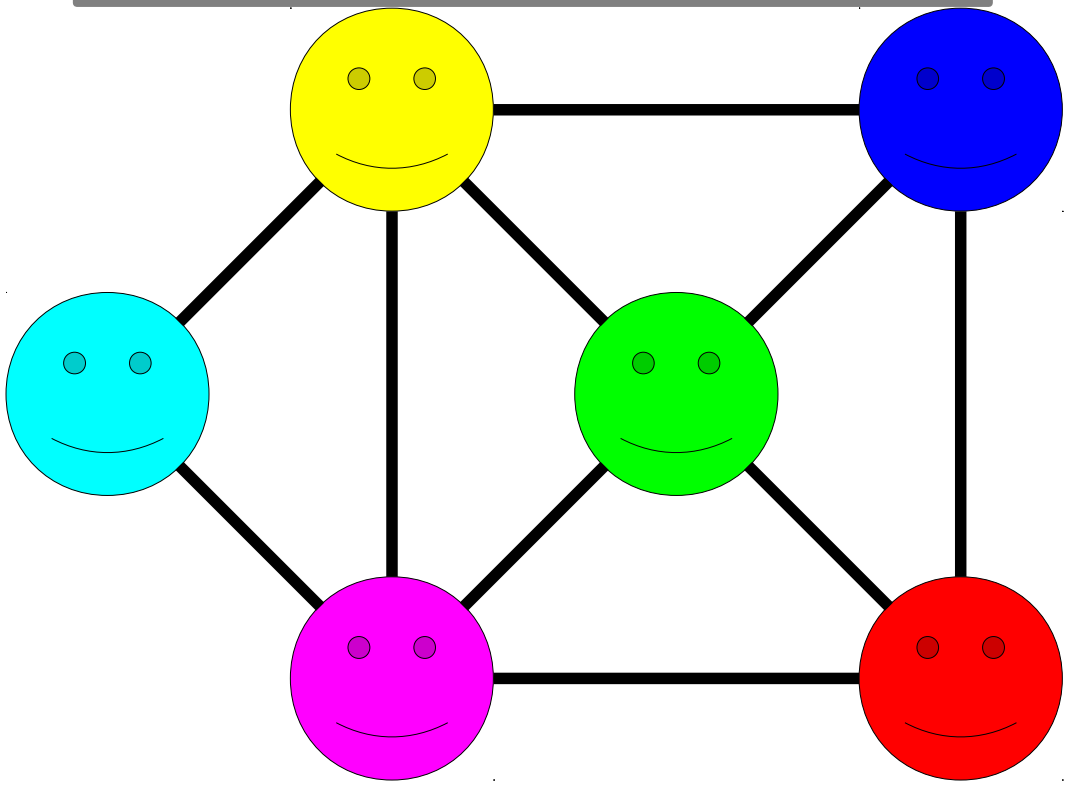


You can think of them as directed graphs with edges both ways.

How can we represent graphs in C++?

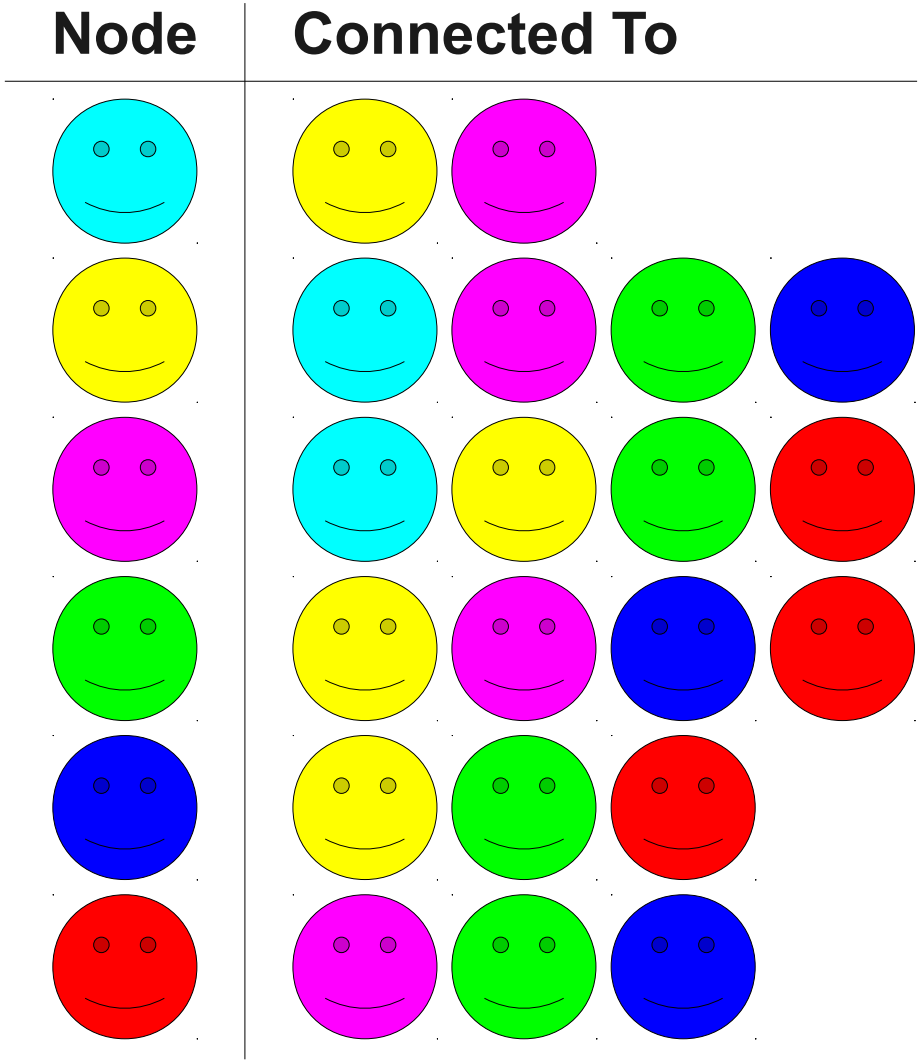
Representing Graphs

We can represent a graph as a map from nodes to the list of nodes each node is connected to.



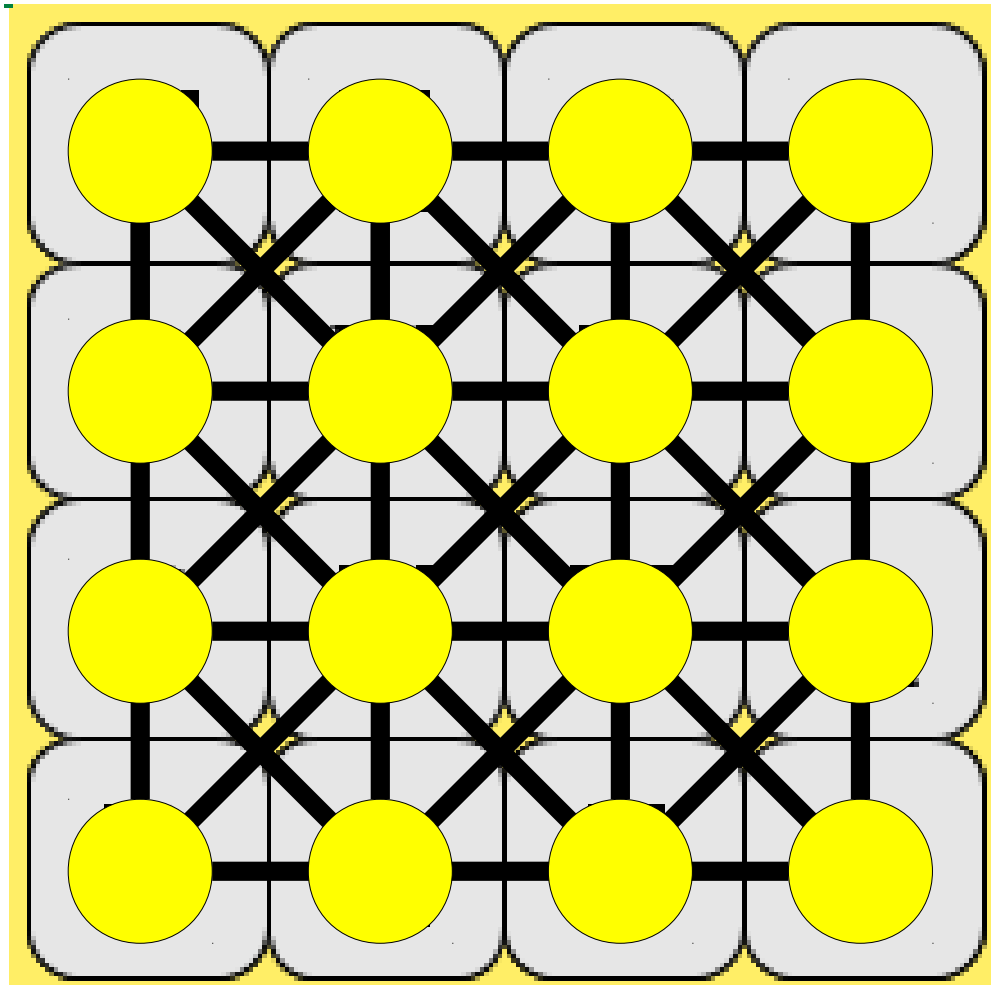
`Map<`*Node**`, Vector<`*Node**`>>`

*Node** `Vector<`*Node**`>`

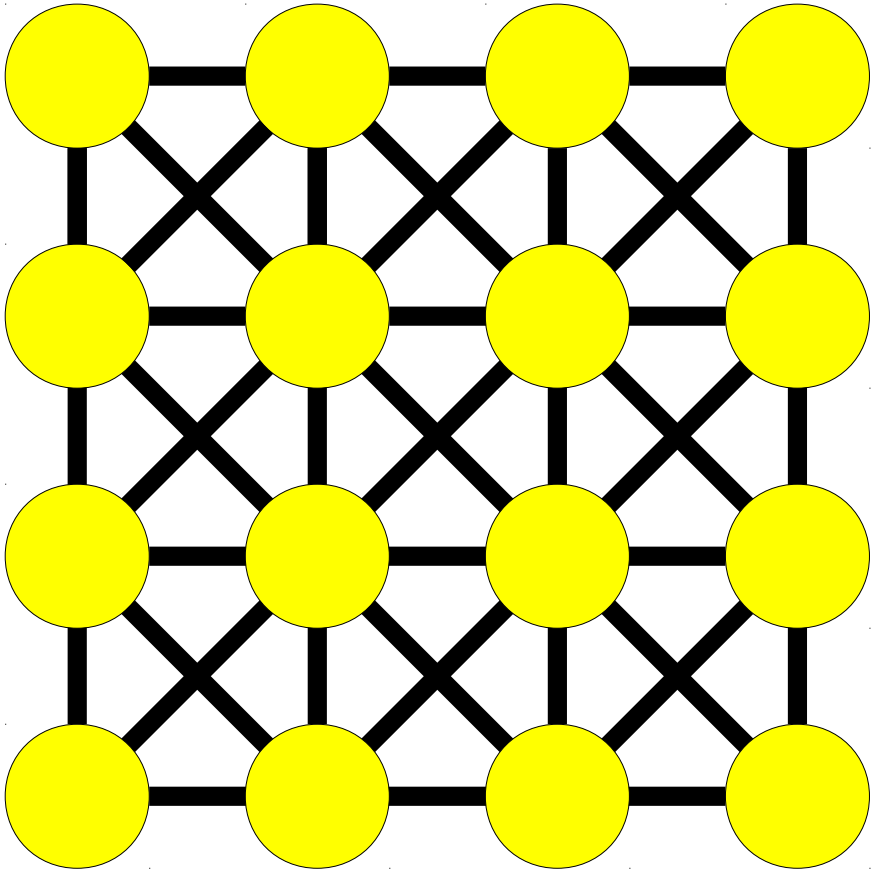


Other Representations

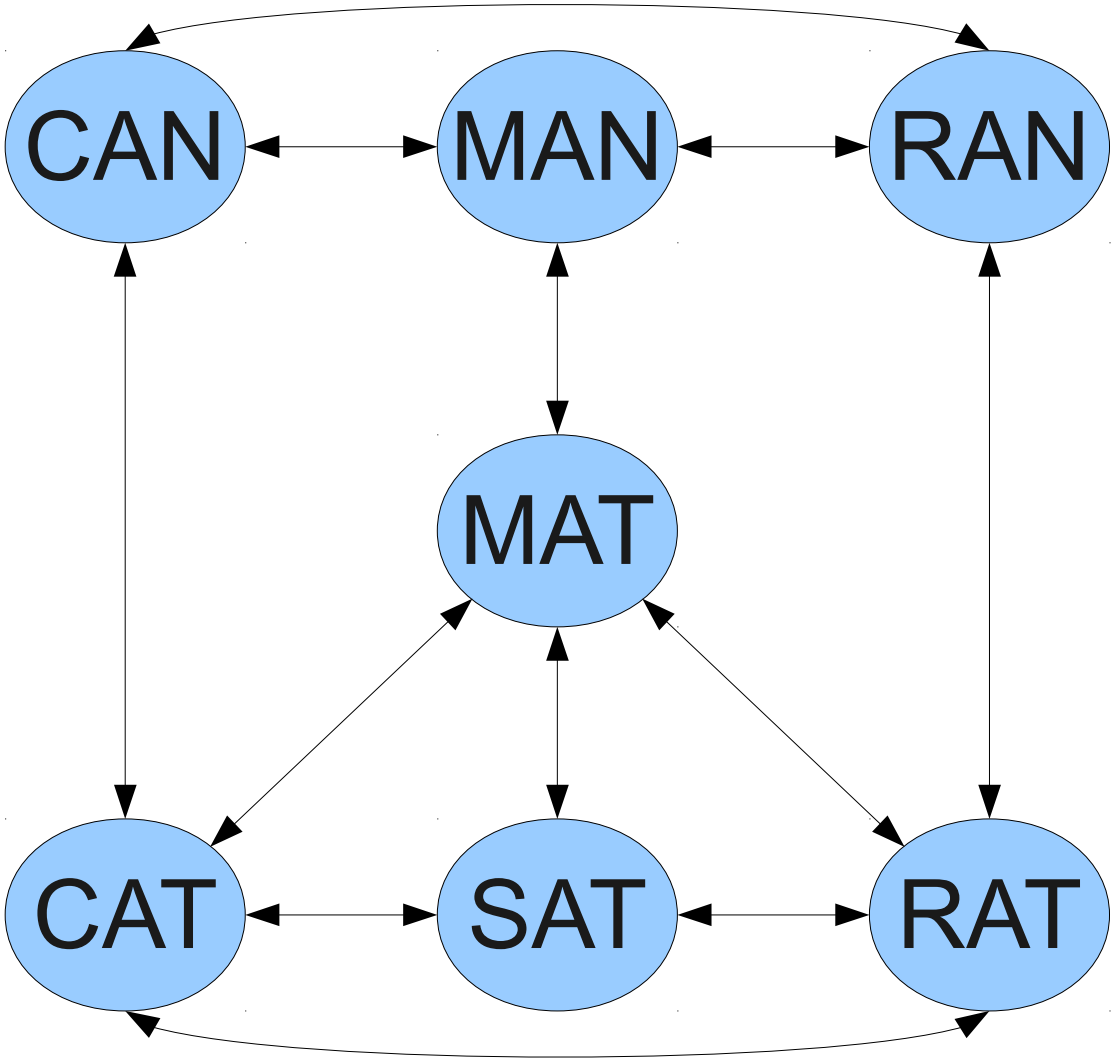
Other Representations



Other Representations



Other Representations



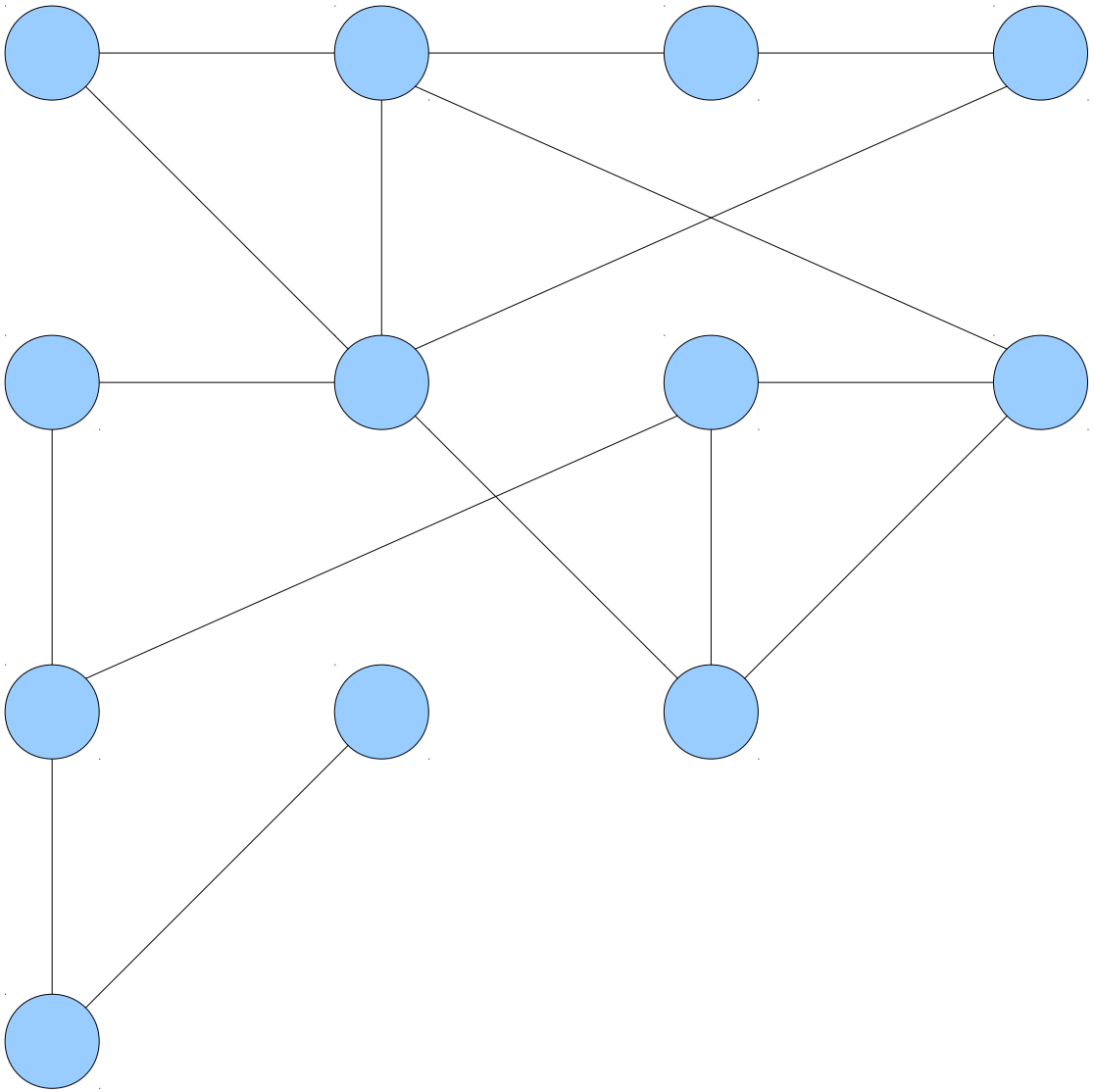
The Wikipedia Graph

- Wikipedia (and the web in general) is a graph!
- Each page is a node.
- There is an edge from one page to another if the first page links to the second.

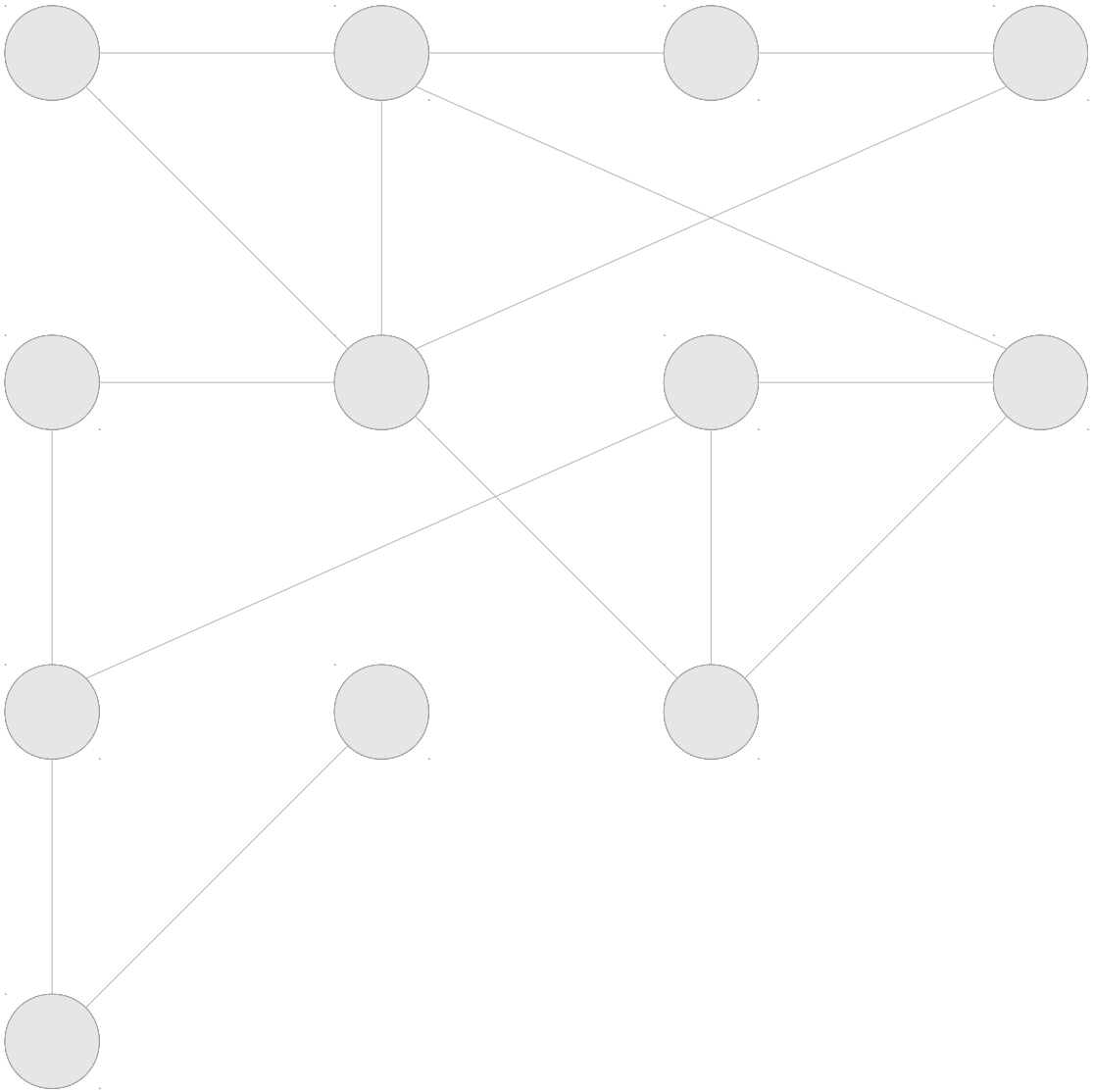
Iterating over a Graph

- Given a linked list, there was just one way to traverse the list.
  - Keep going forward.
- In a binary search tree, there are many traversal strategies:
  - An *inorder* traversal that produces all the elements in sorted order.
  - A *postorder* traversal used to delete all the nodes in the BST.
- There are *many* ways to iterate over a graph.

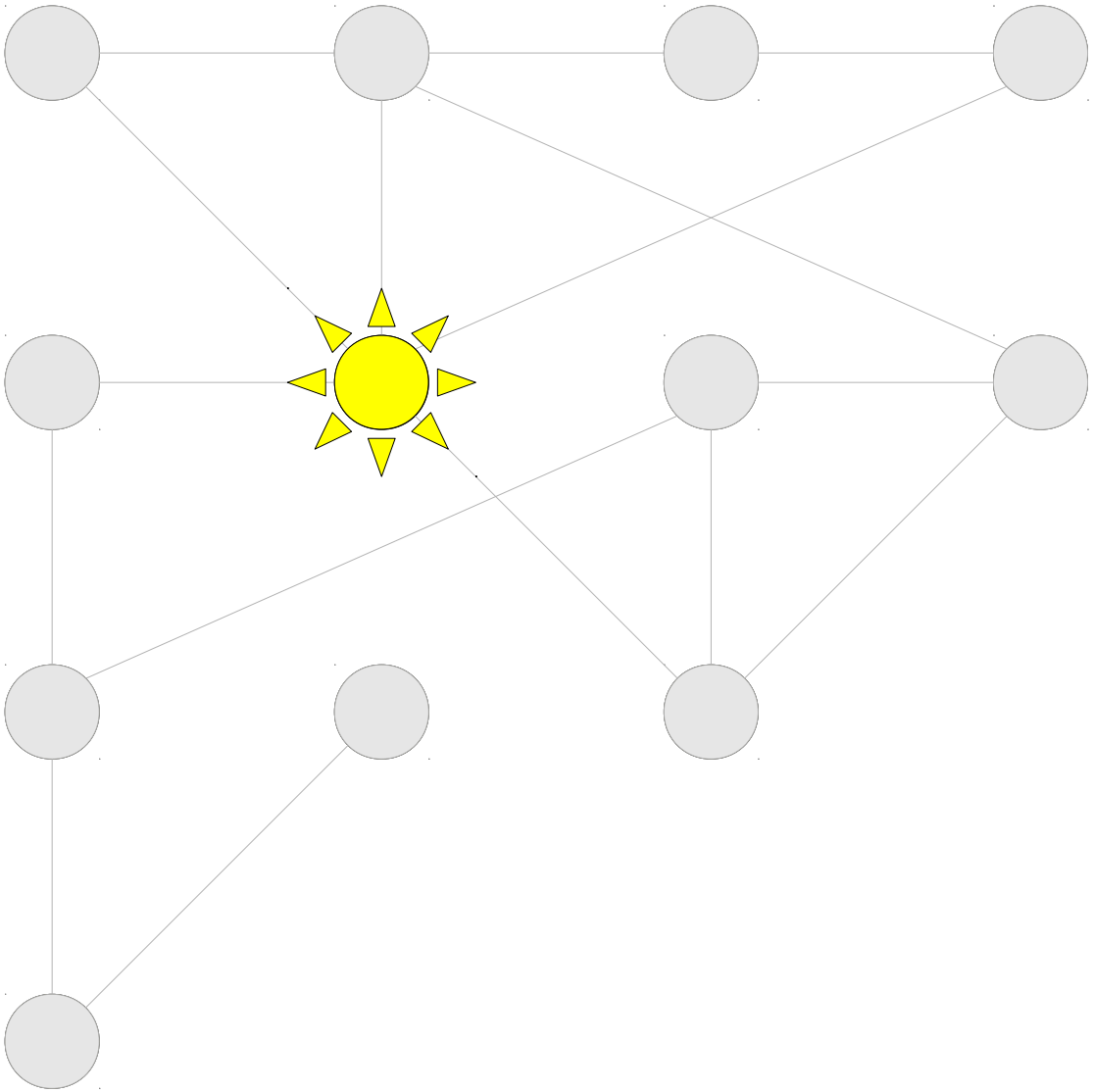
Depth-First Search



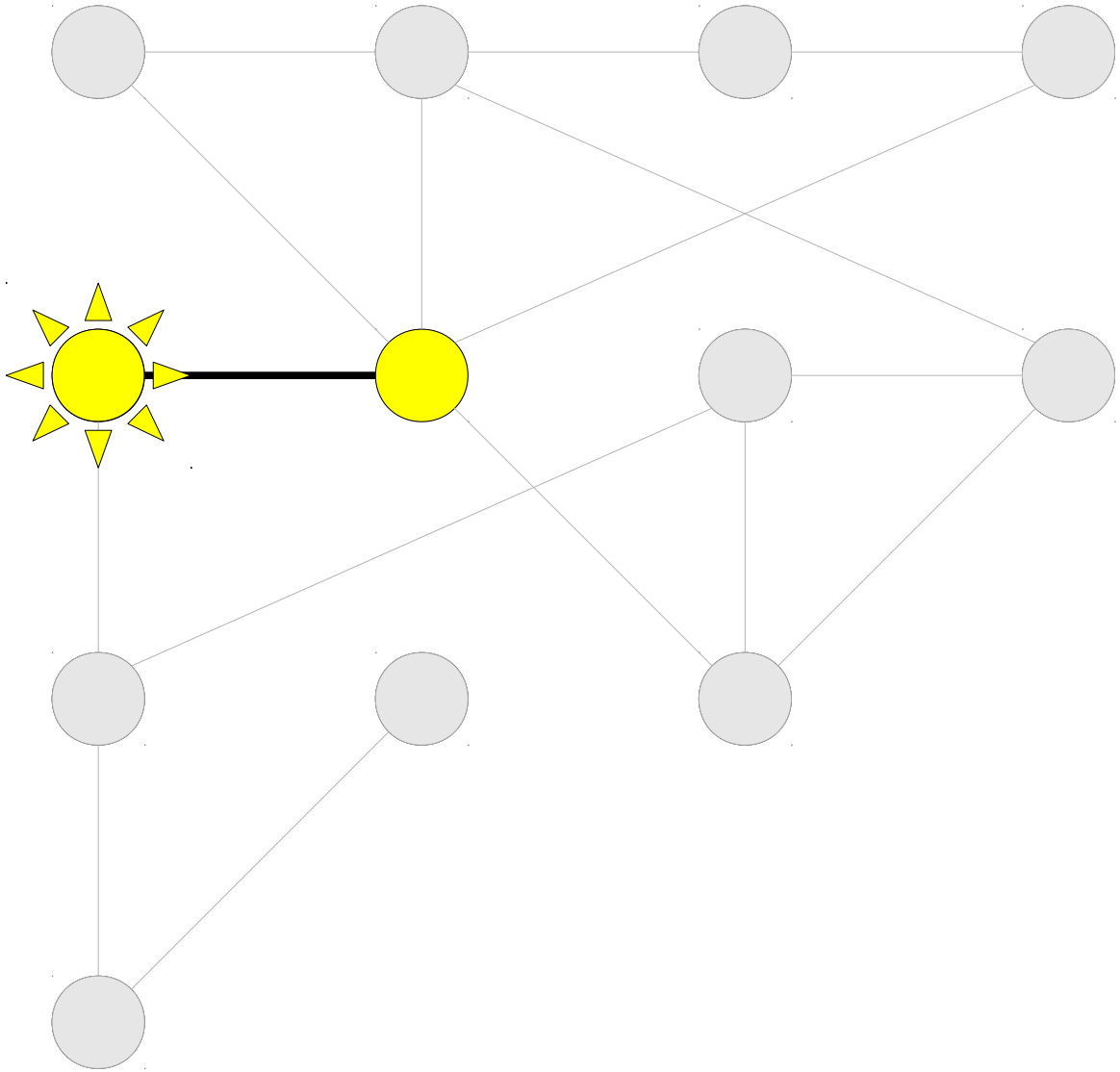
Depth-First Search



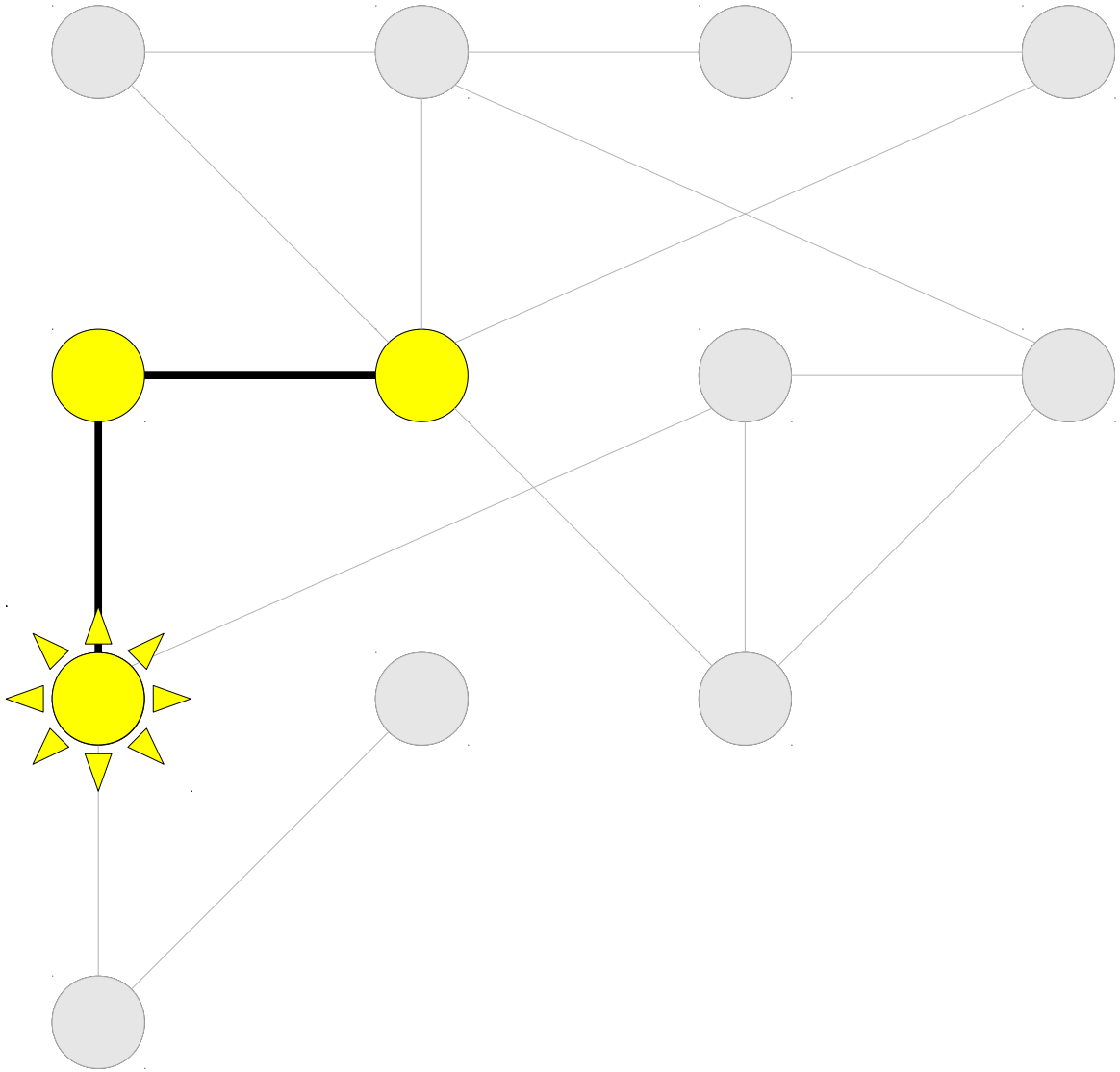
Depth-First Search



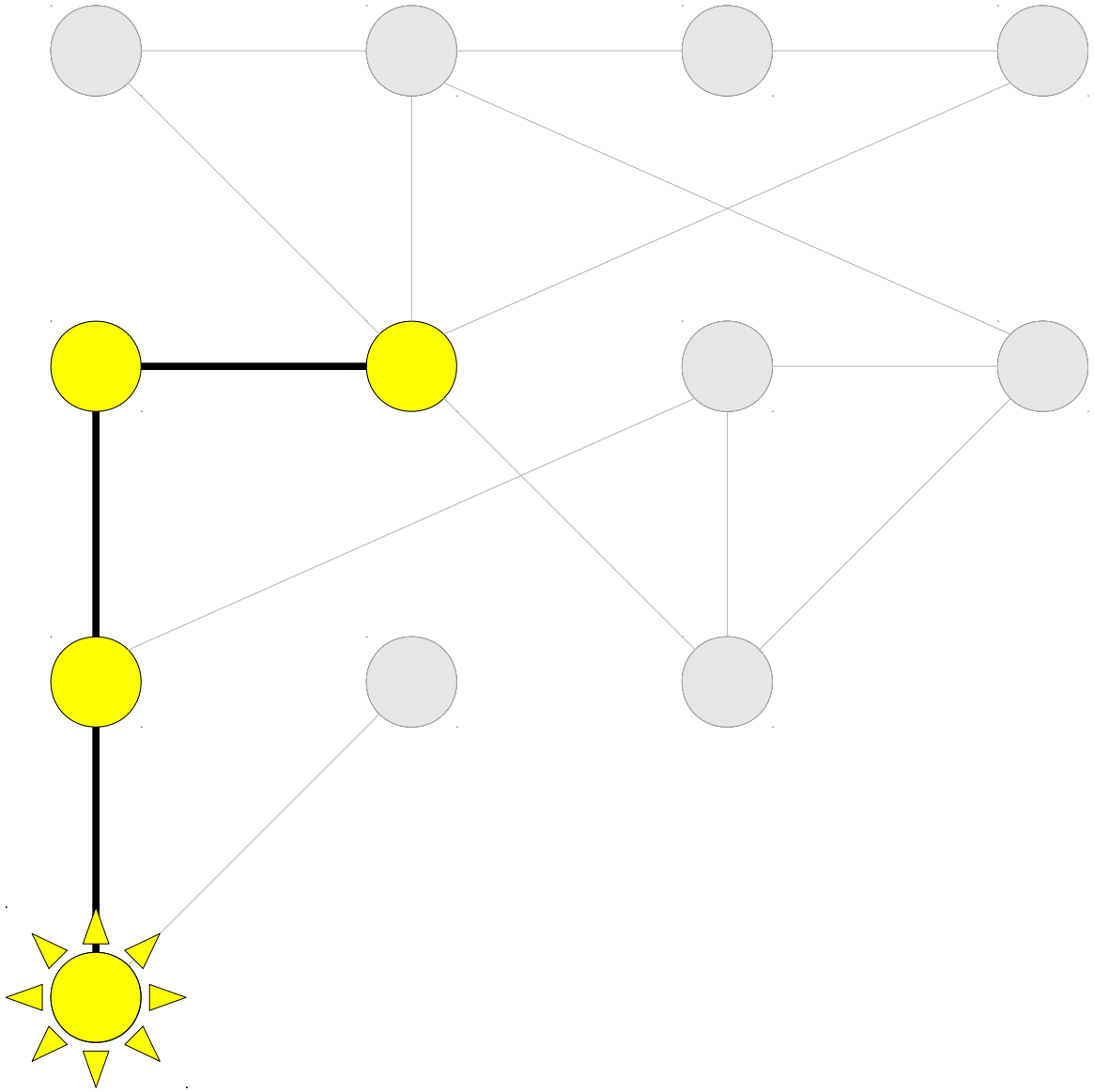
Depth-First Search



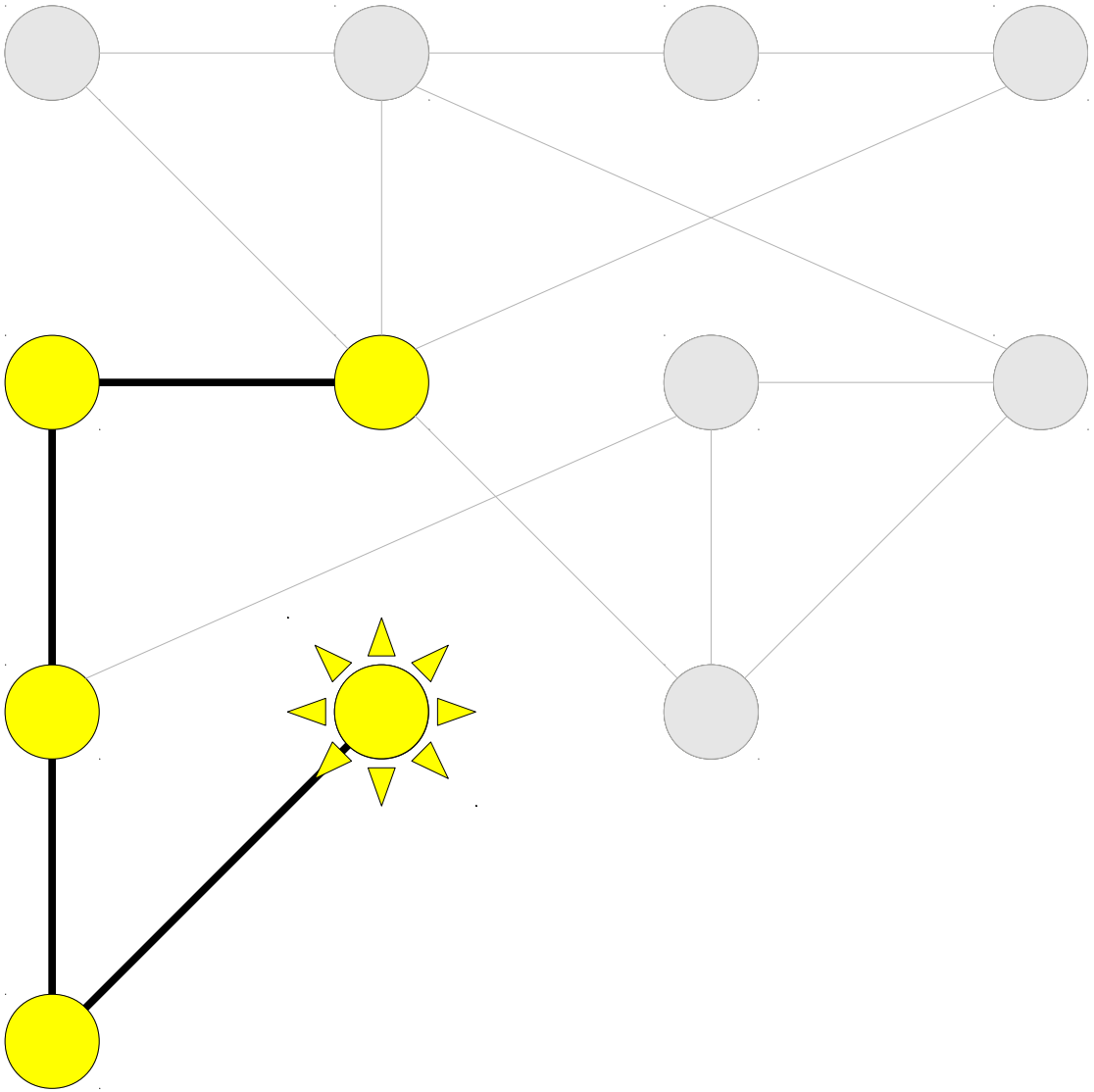
Depth-First Search



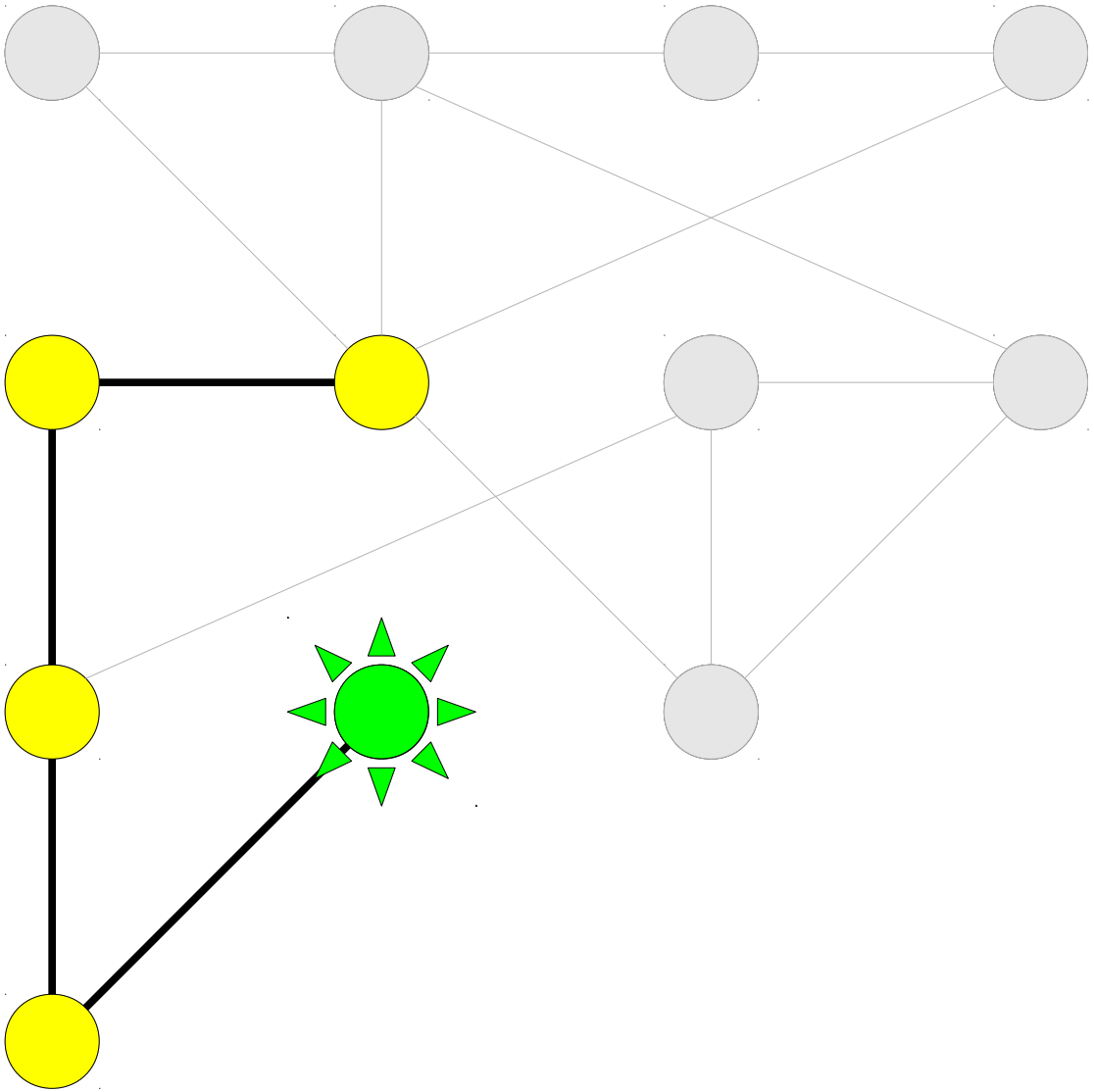
Depth-First Search



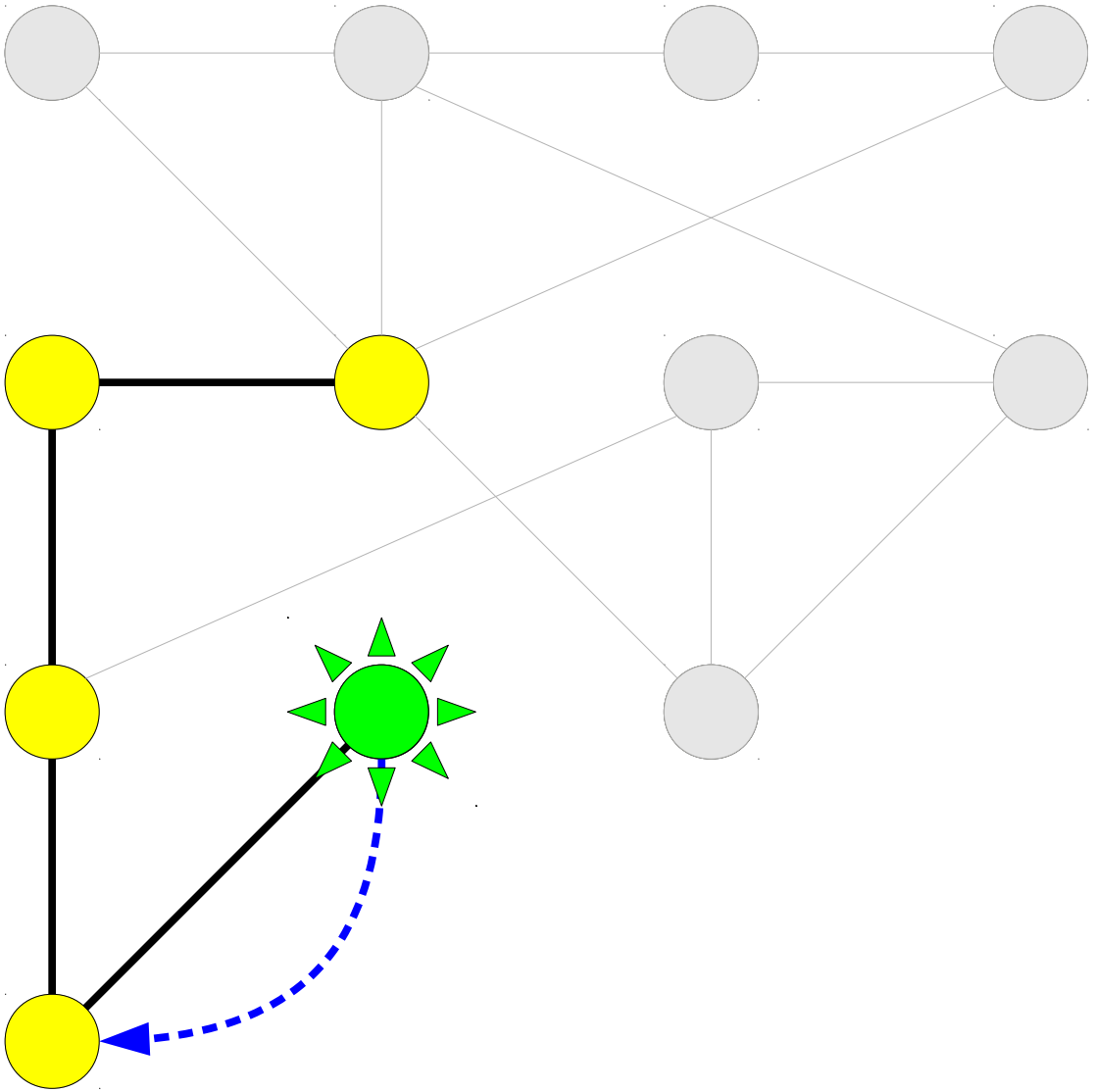
Depth-First Search



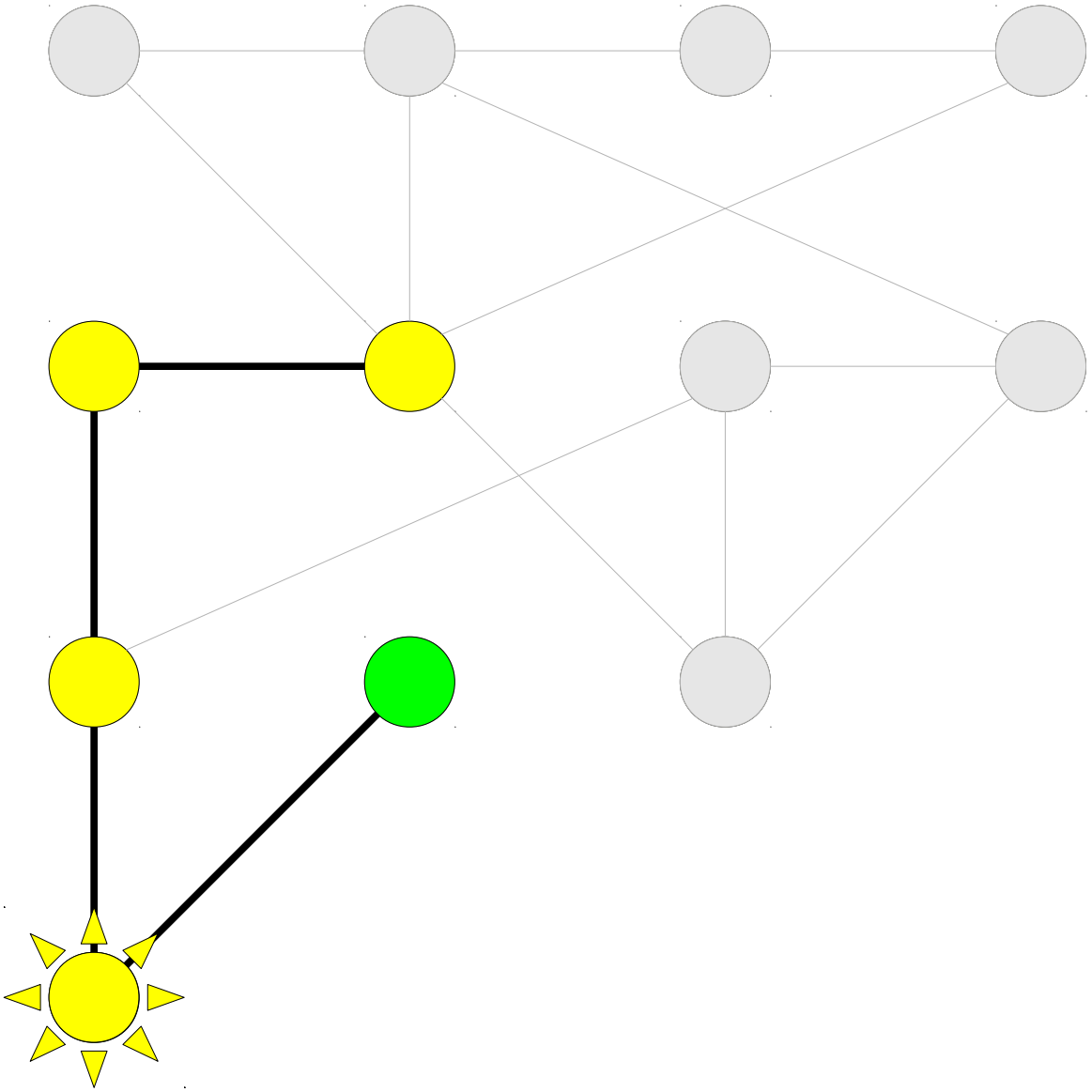
Depth-First Search



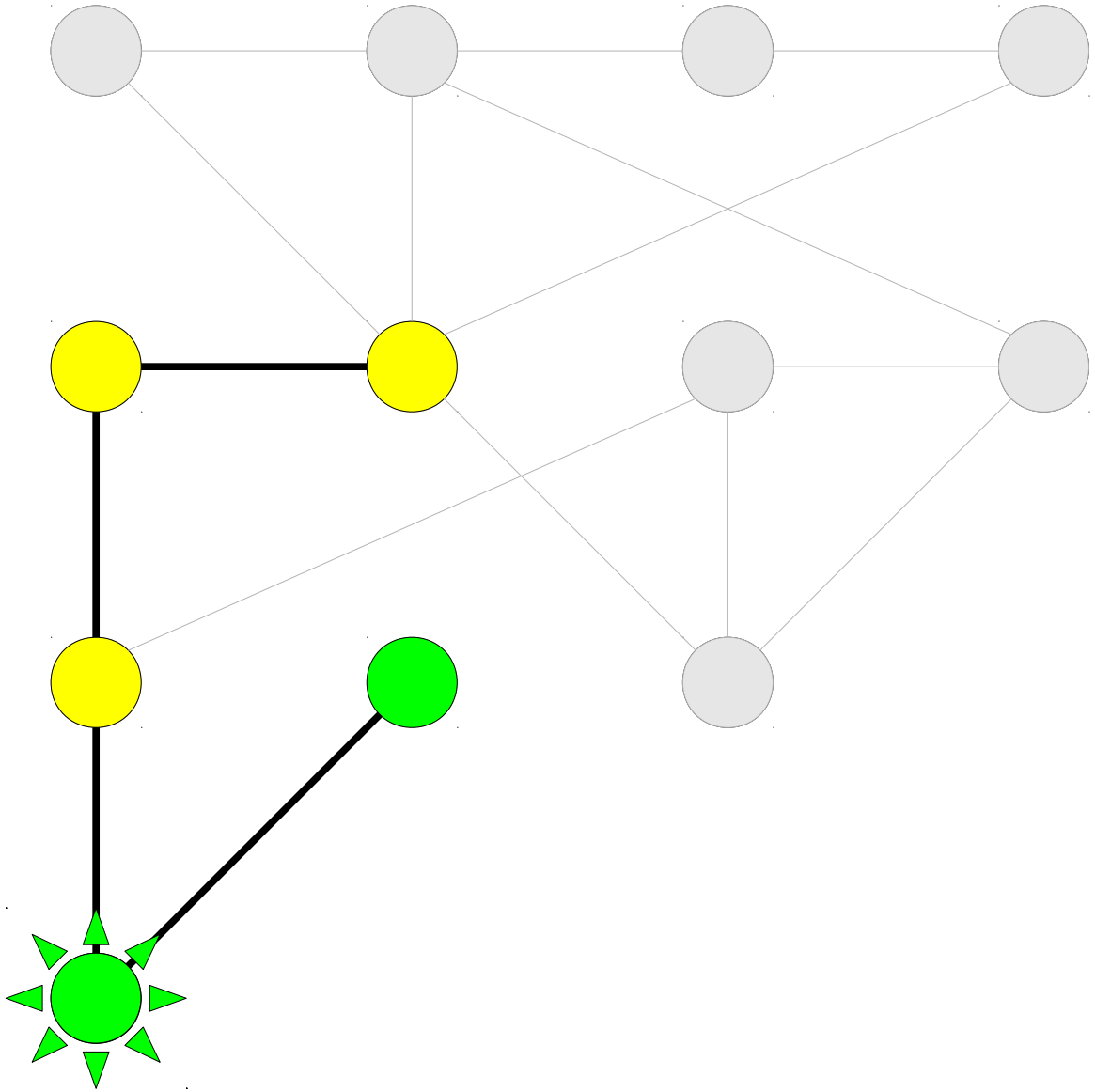
Depth-First Search



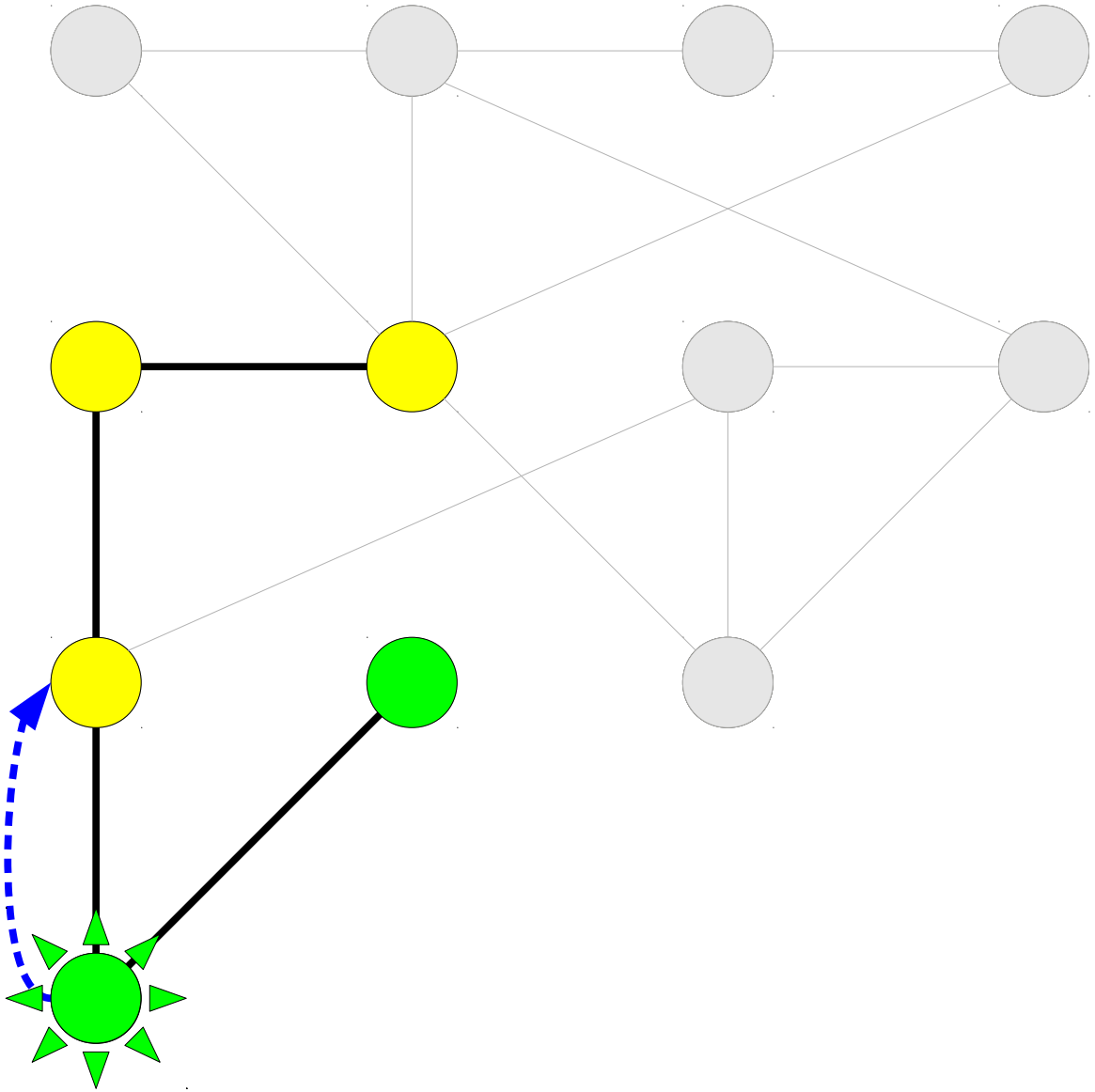
Depth-First Search



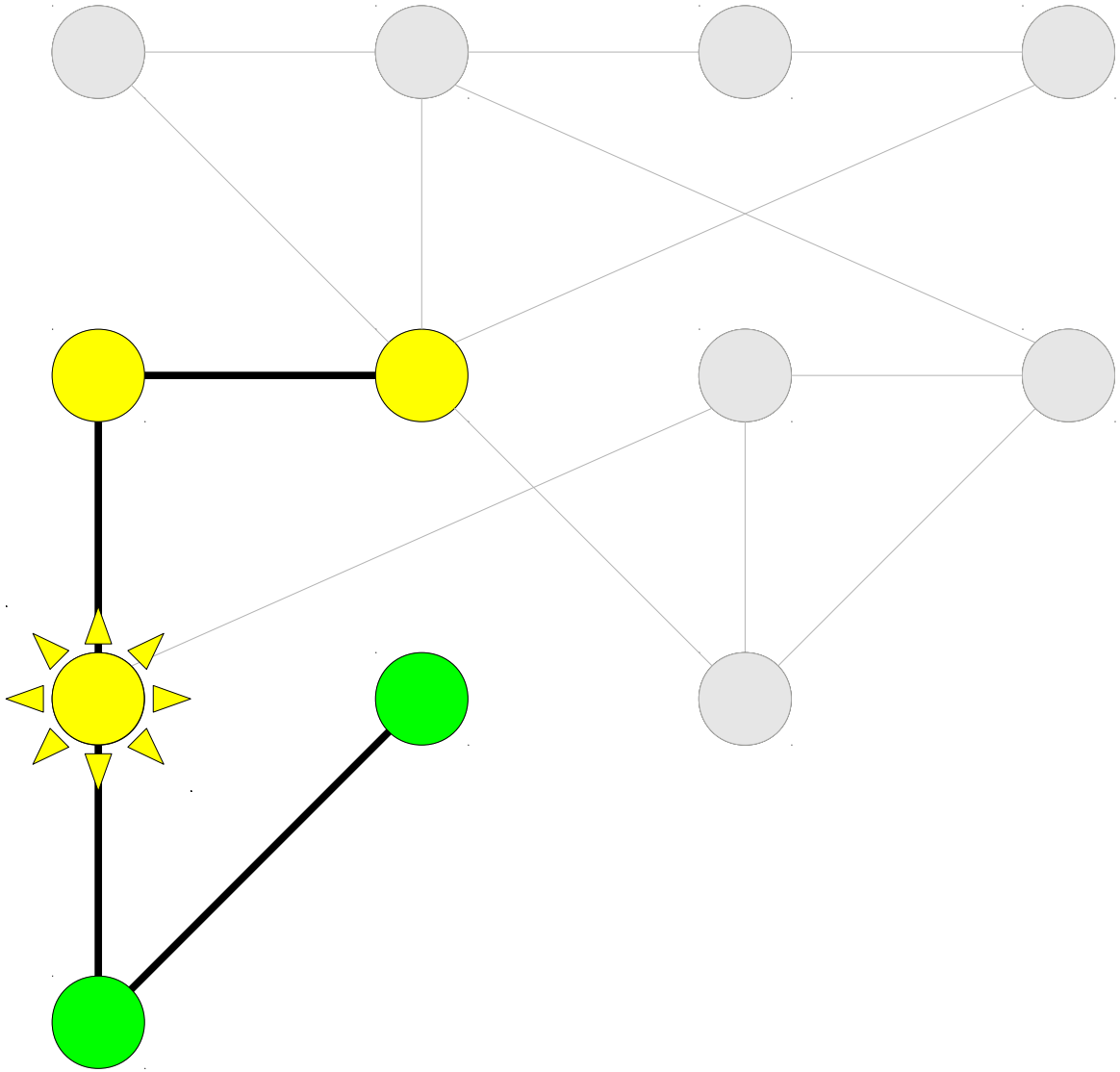
Depth-First Search



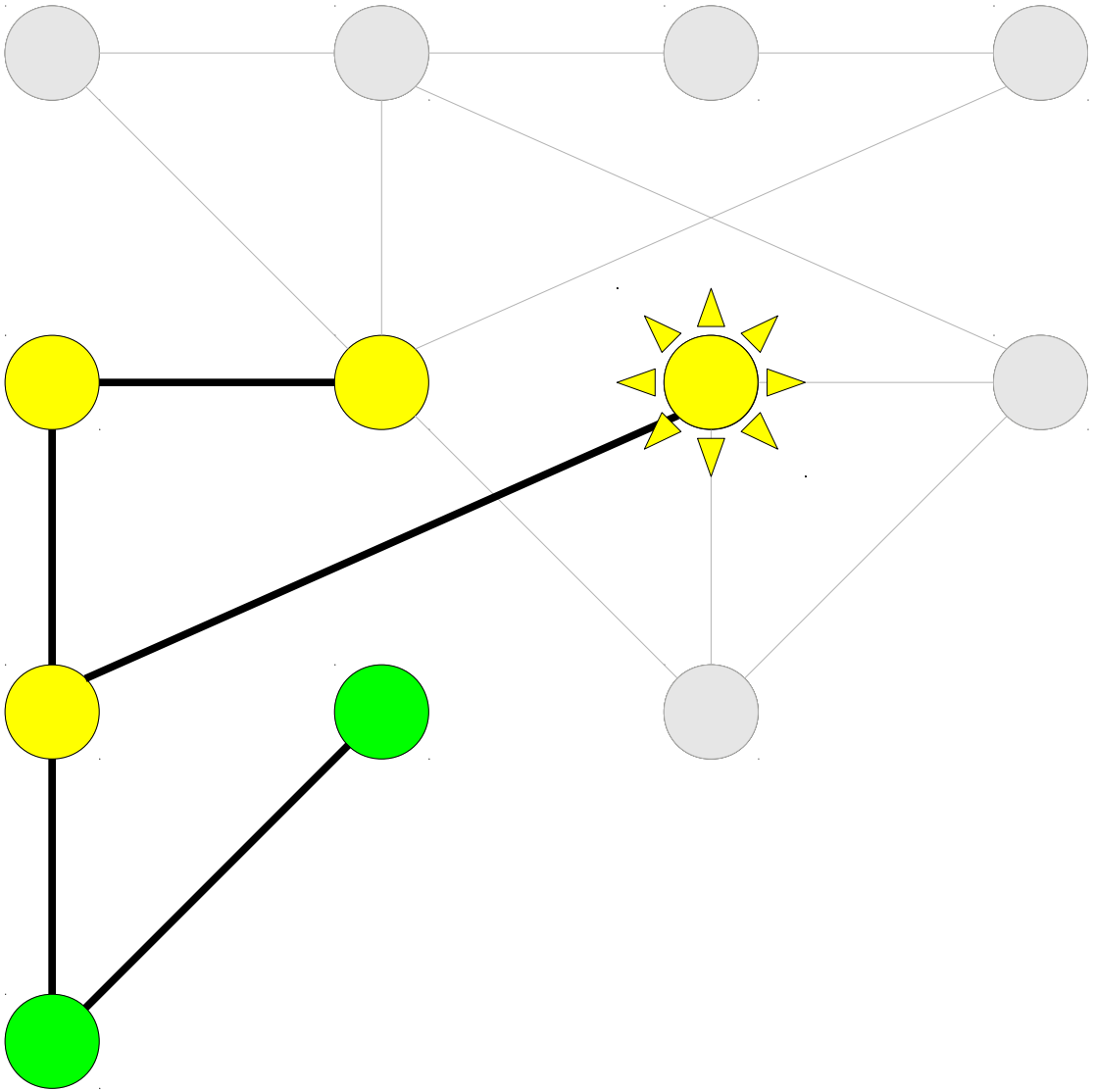
Depth-First Search



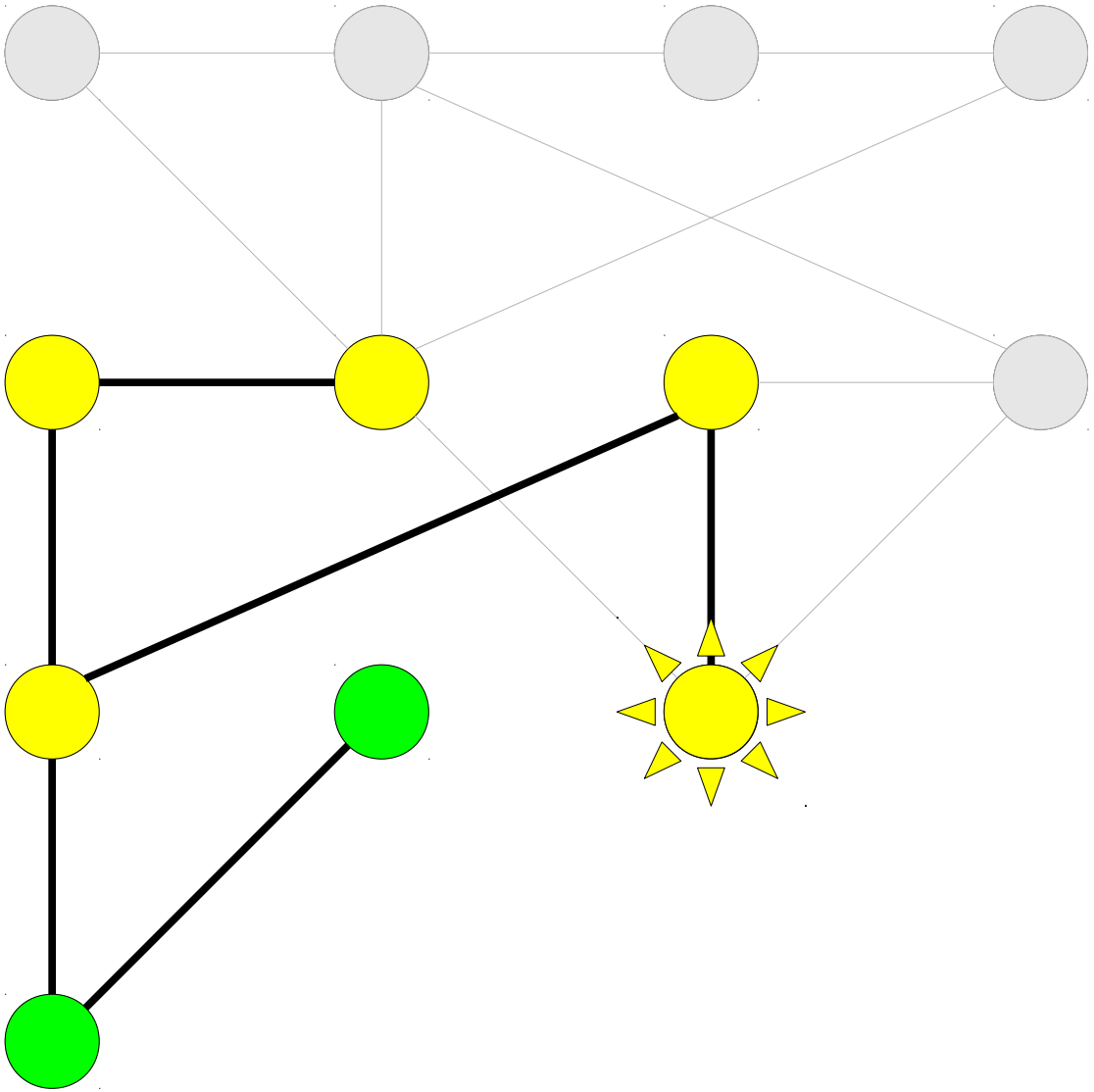
Depth-First Search



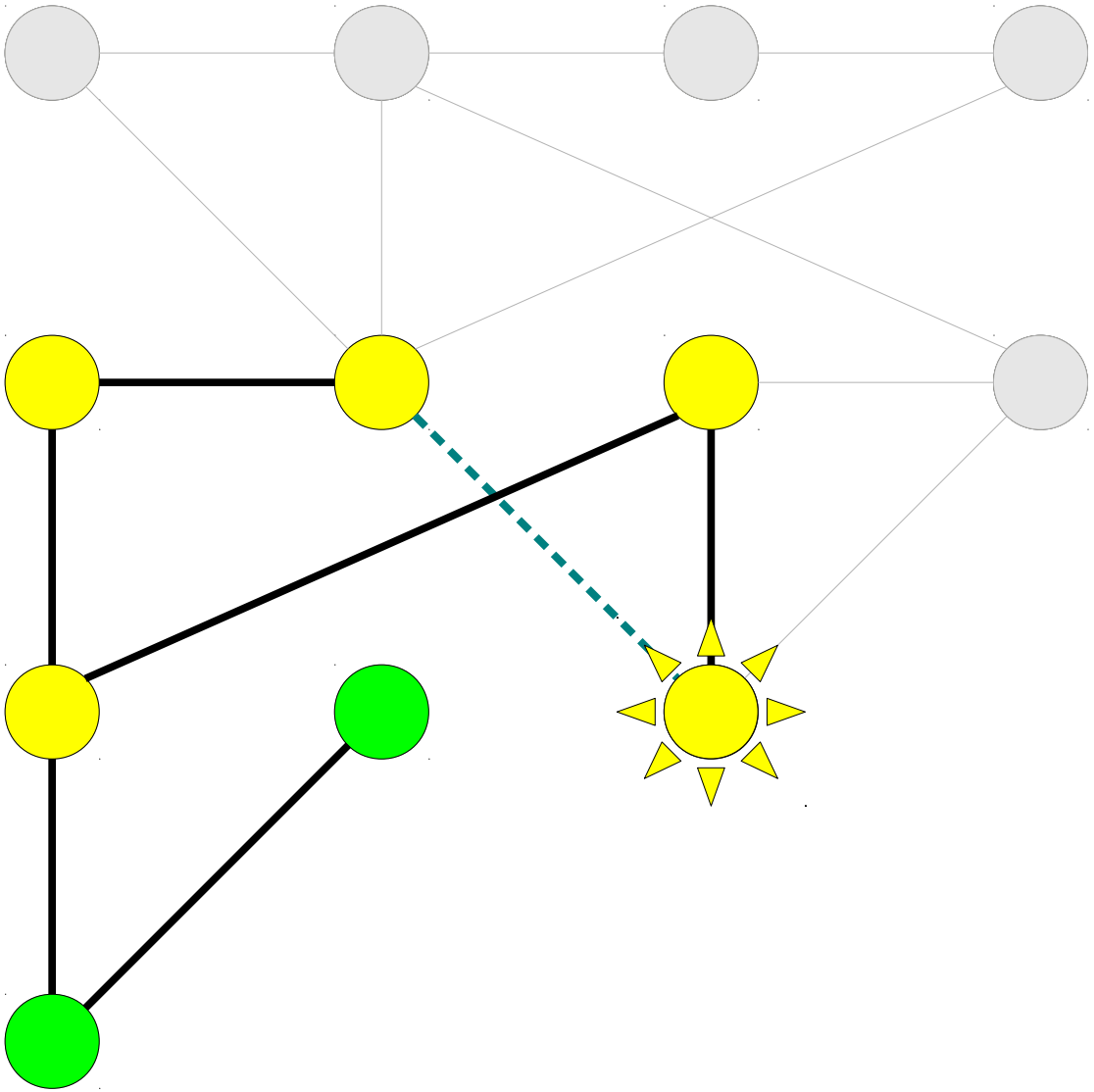
Depth-First Search



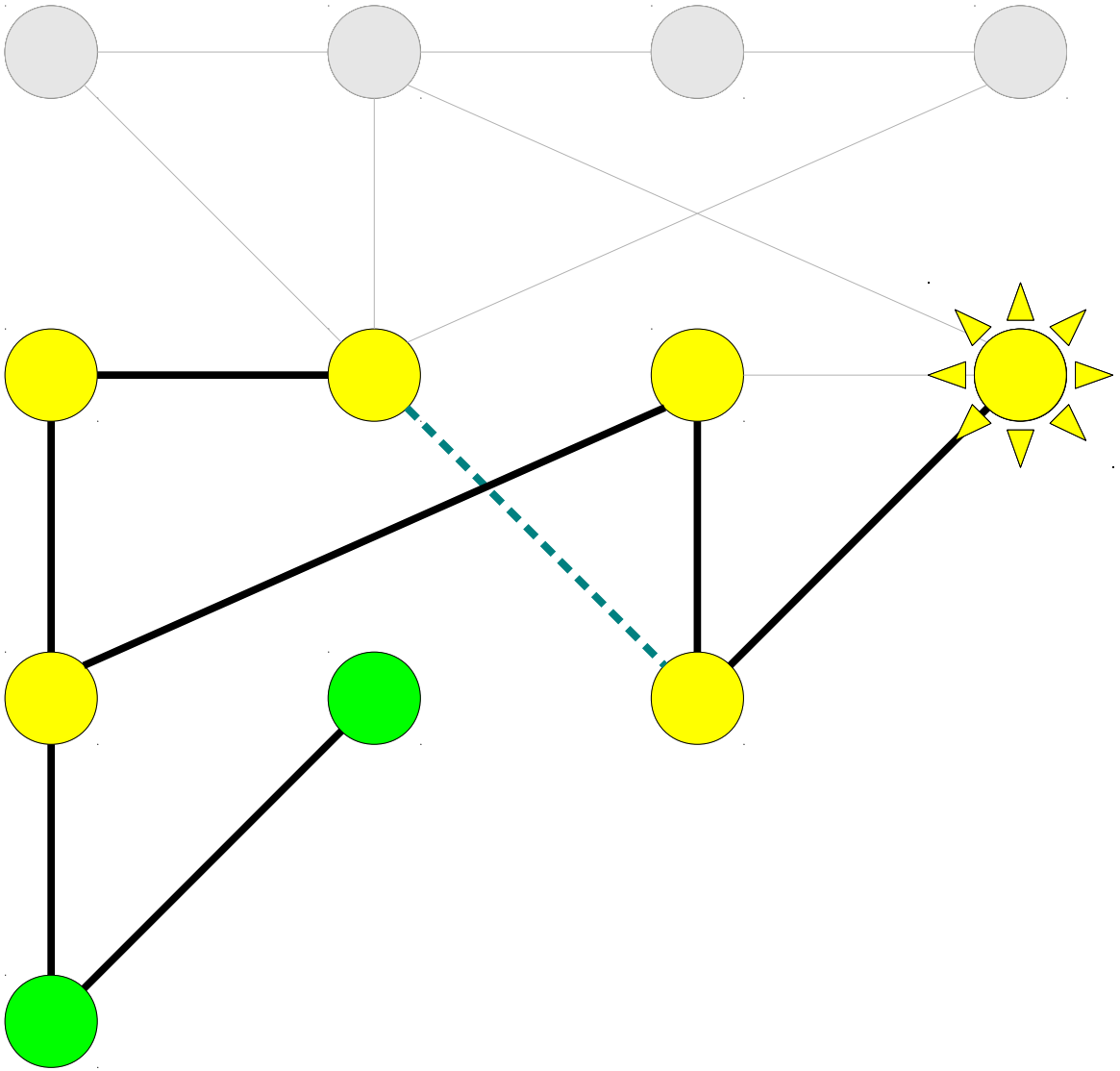
Depth-First Search



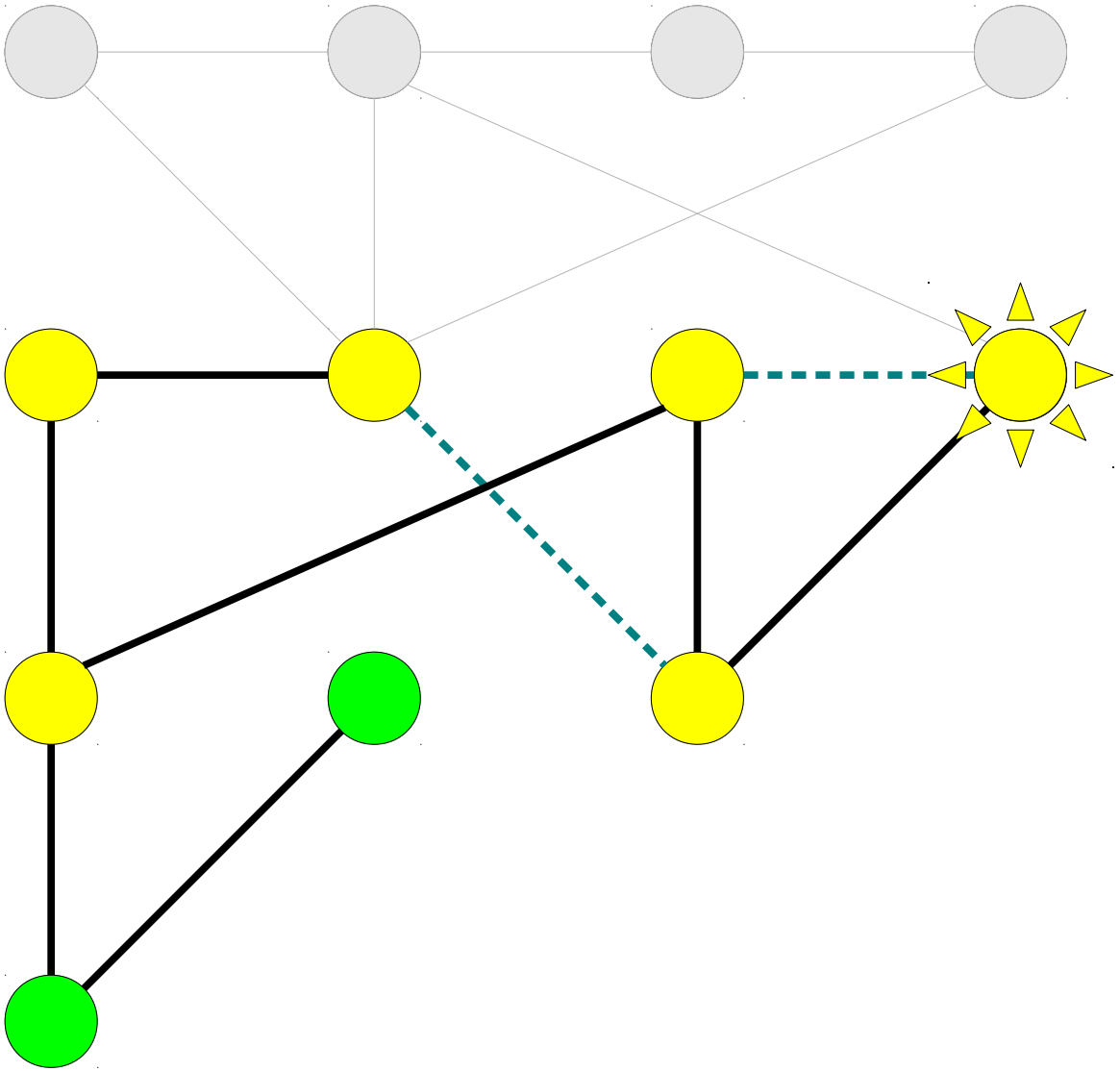
Depth-First Search



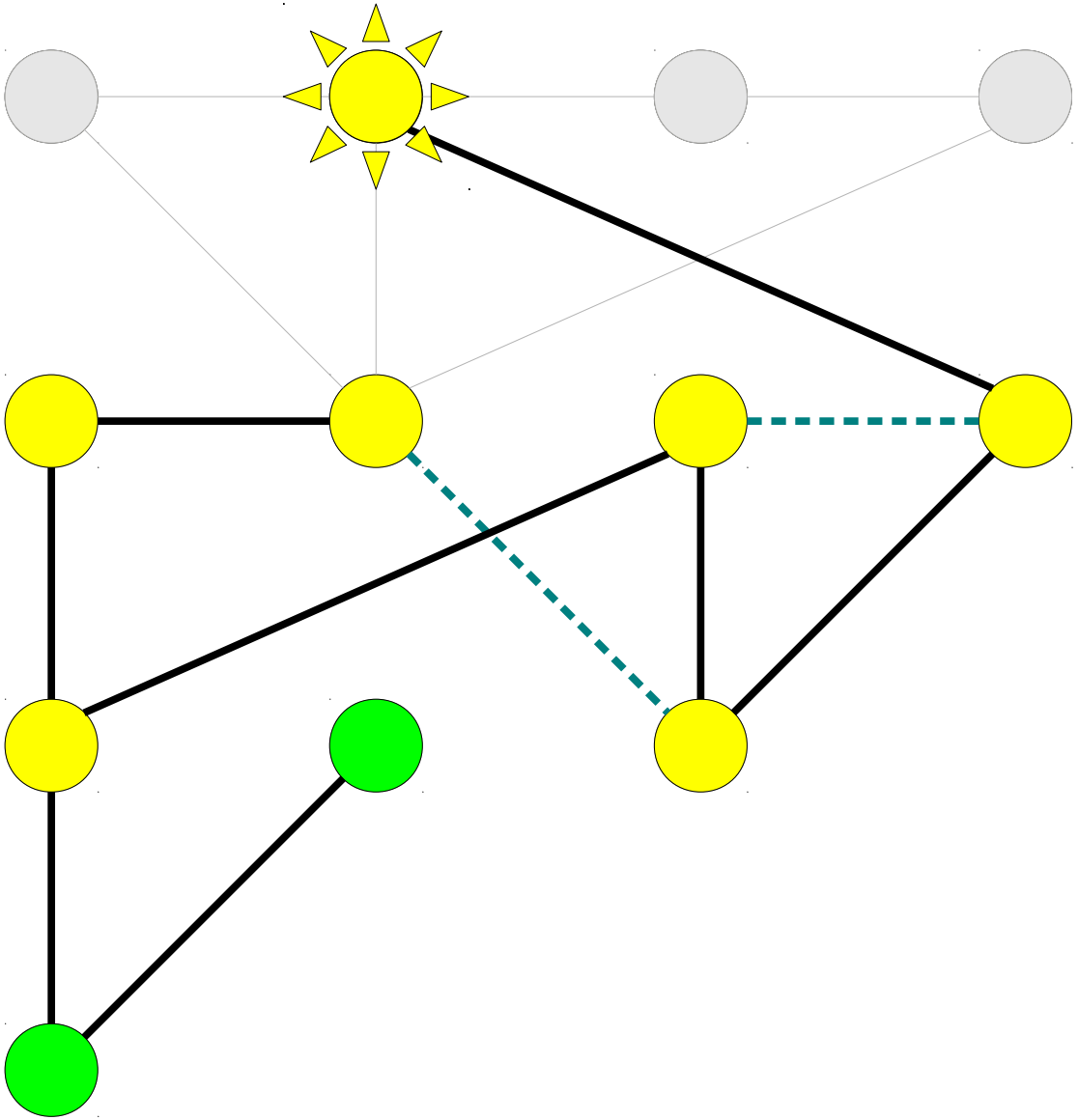
Depth-First Search



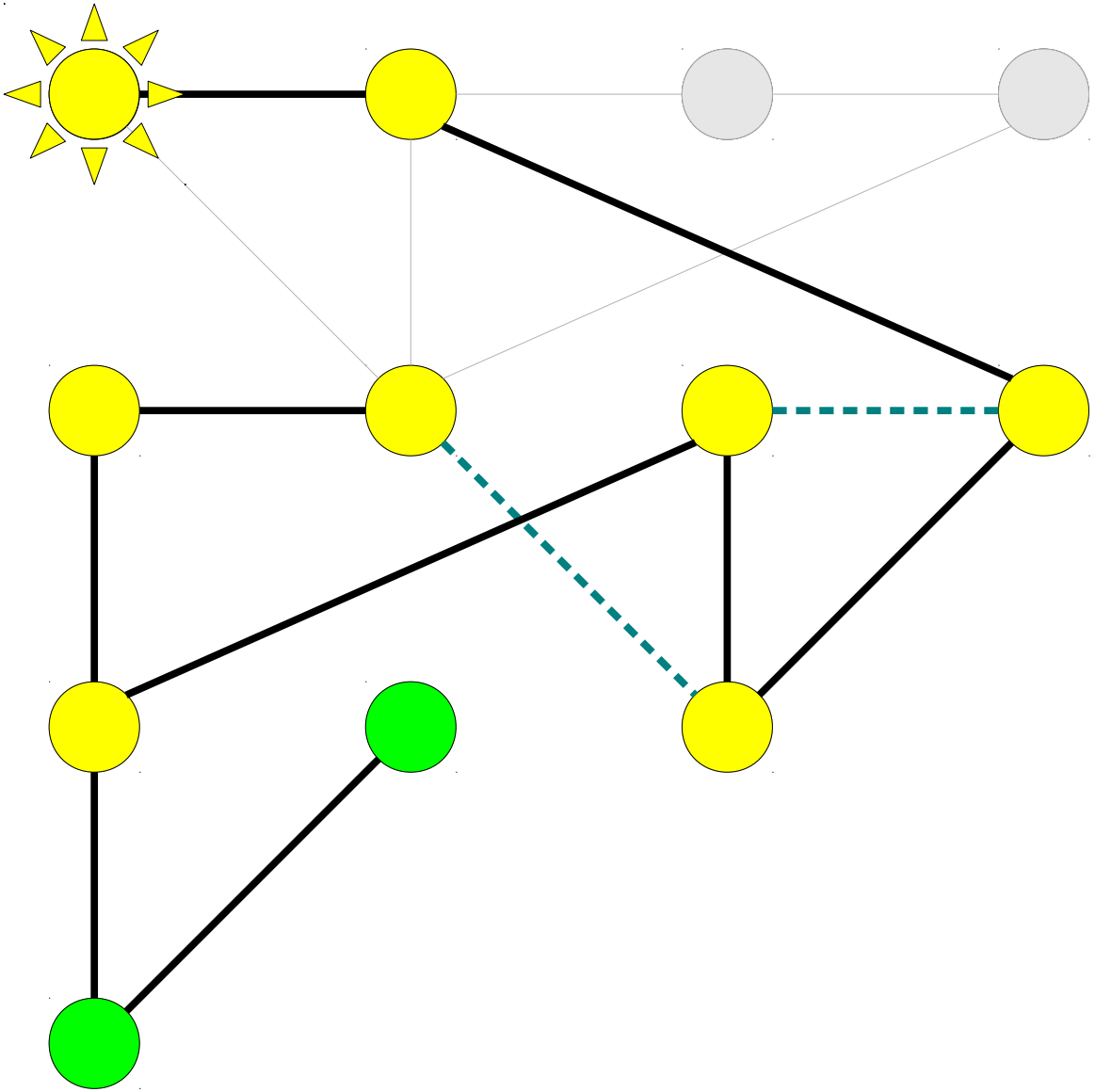
Depth-First Search



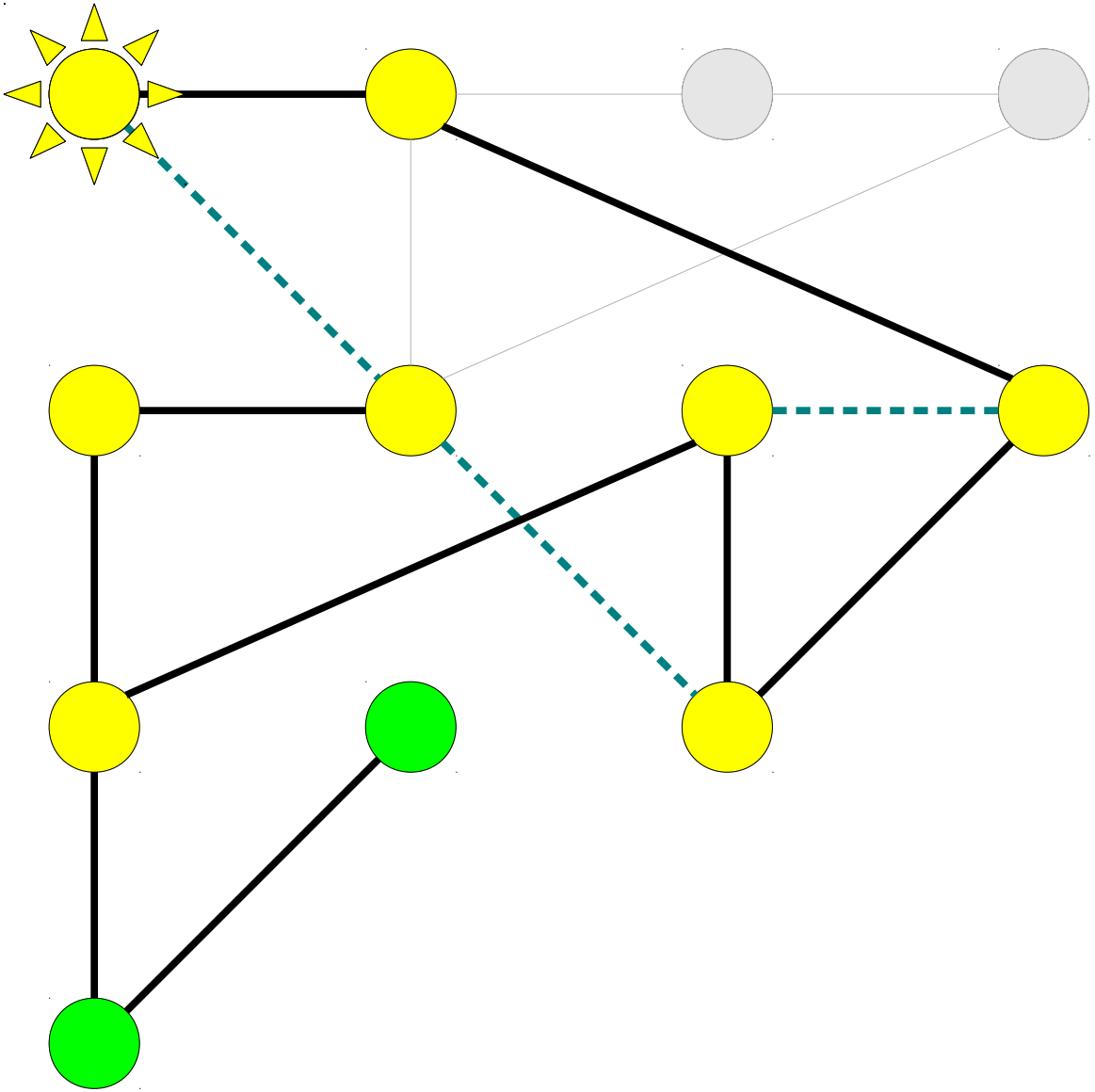
Depth-First Search



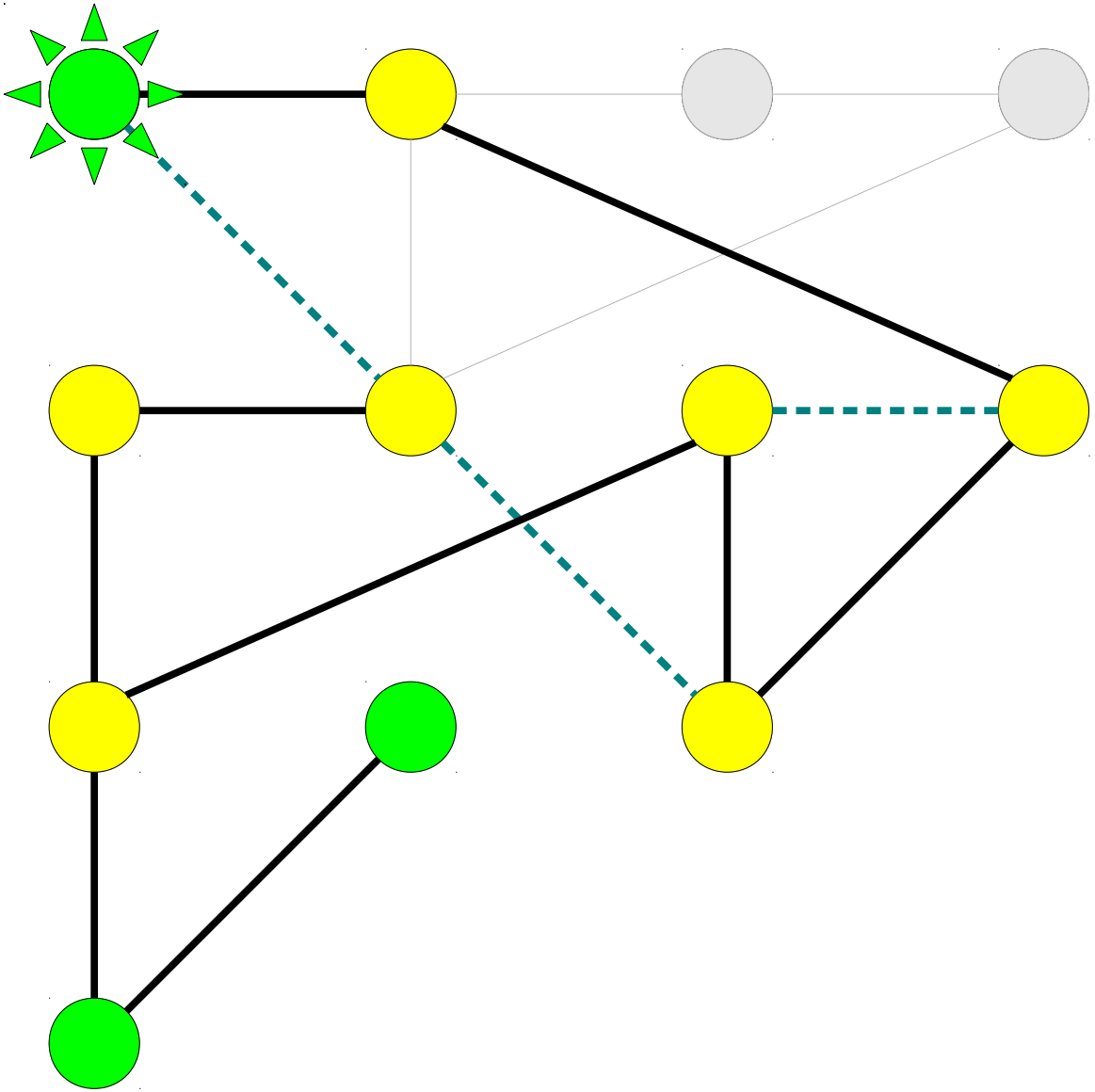
Depth-First Search



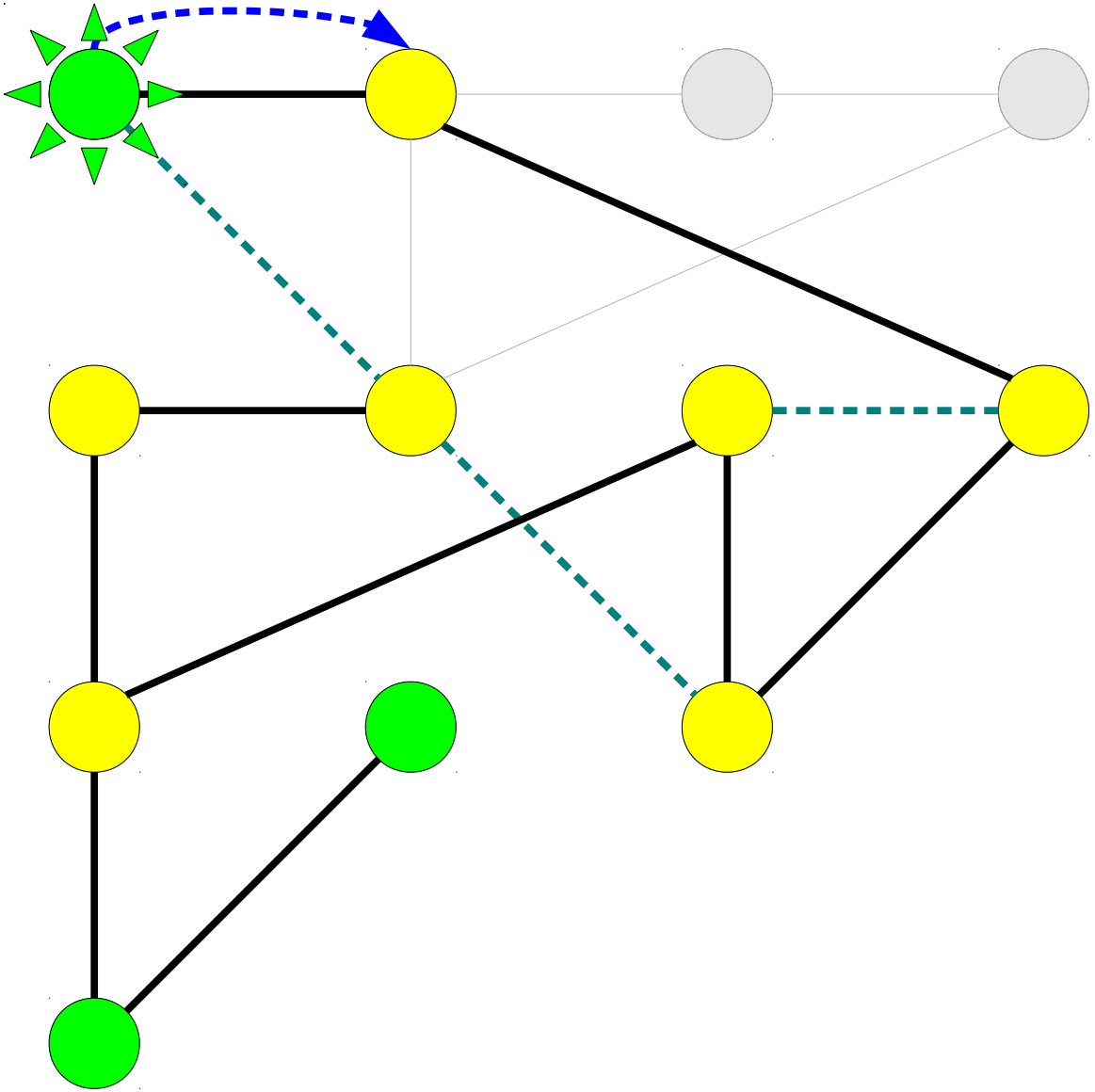
Depth-First Search



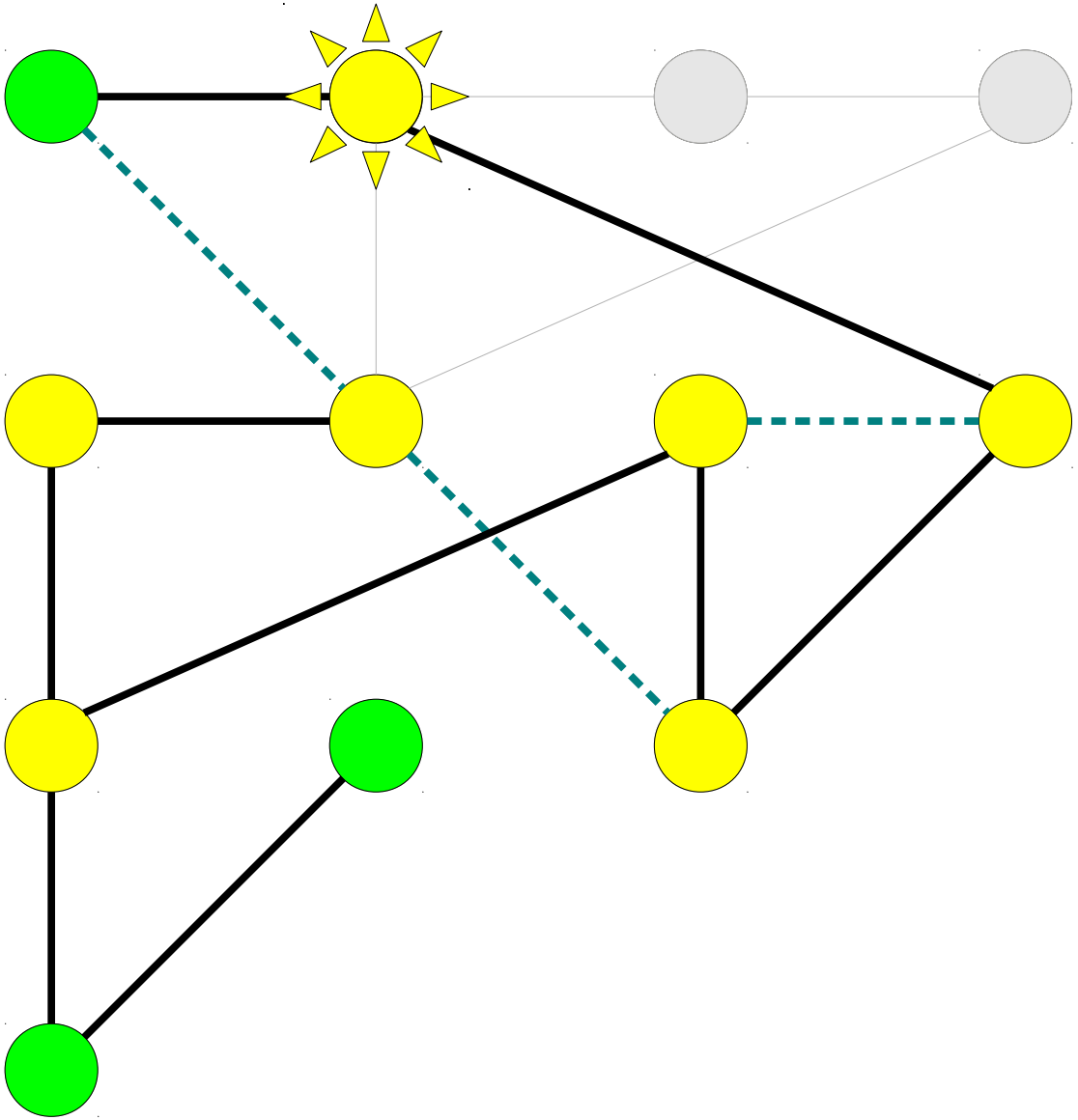
Depth-First Search



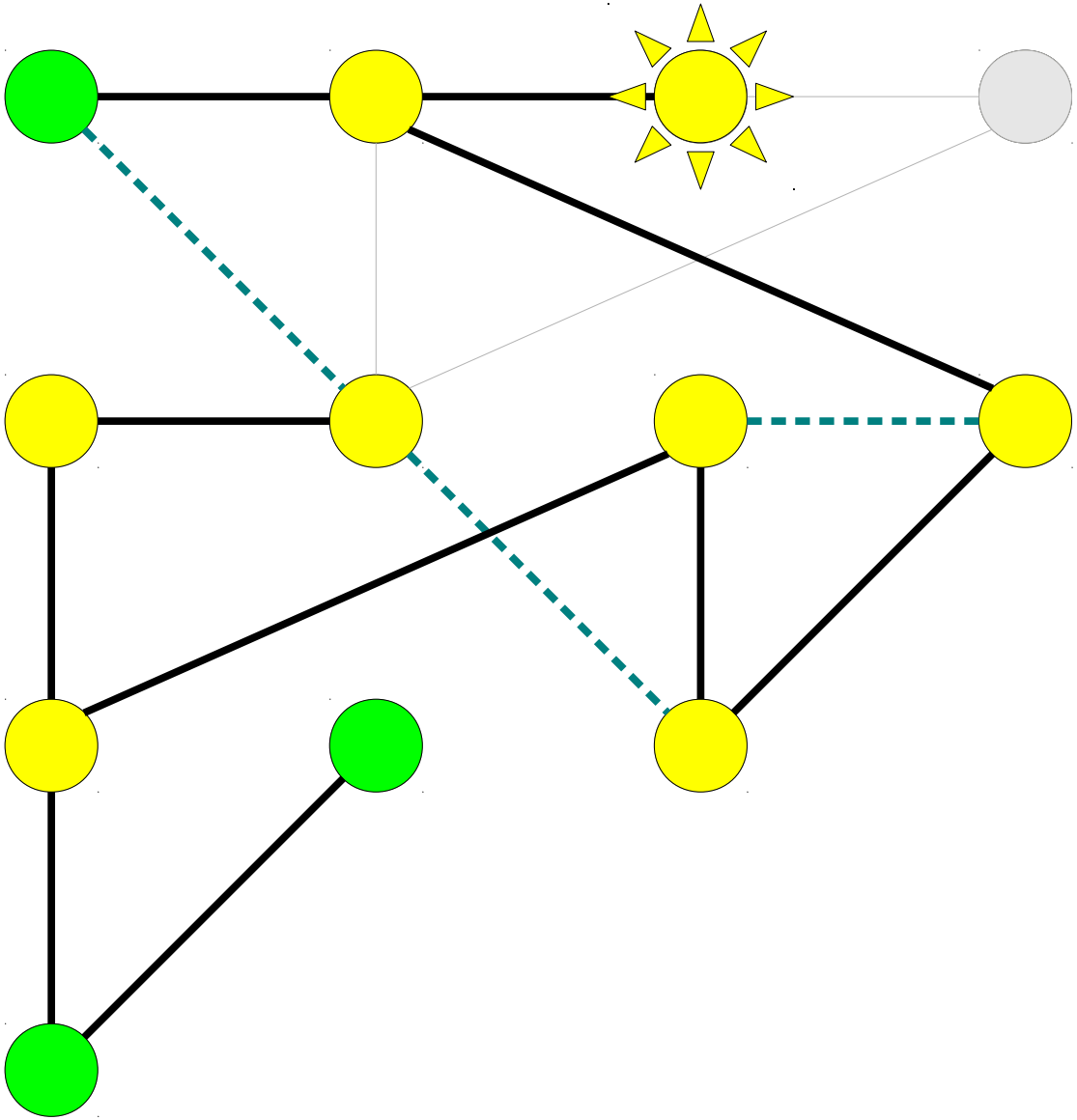
Depth-First Search



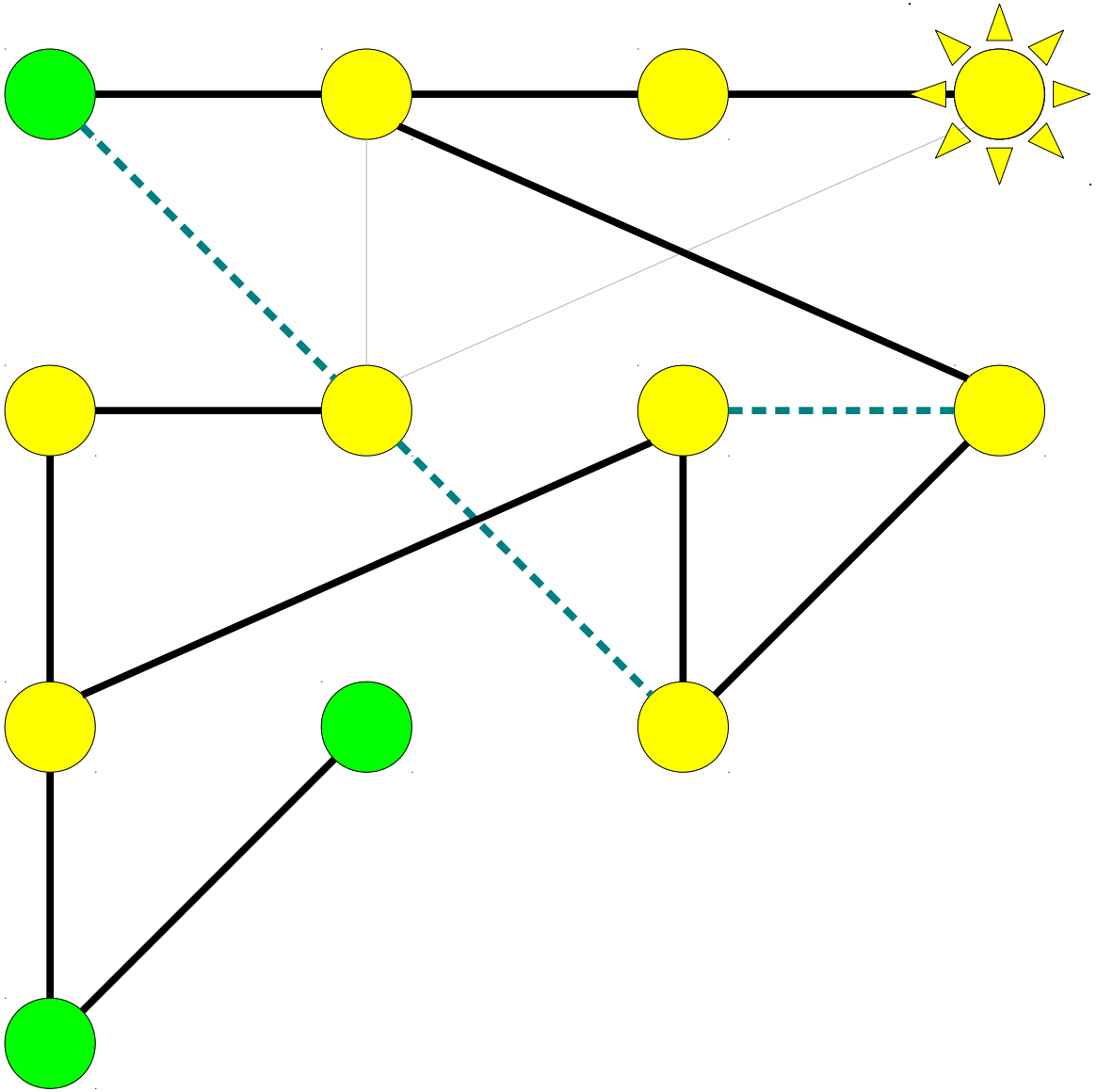
Depth-First Search



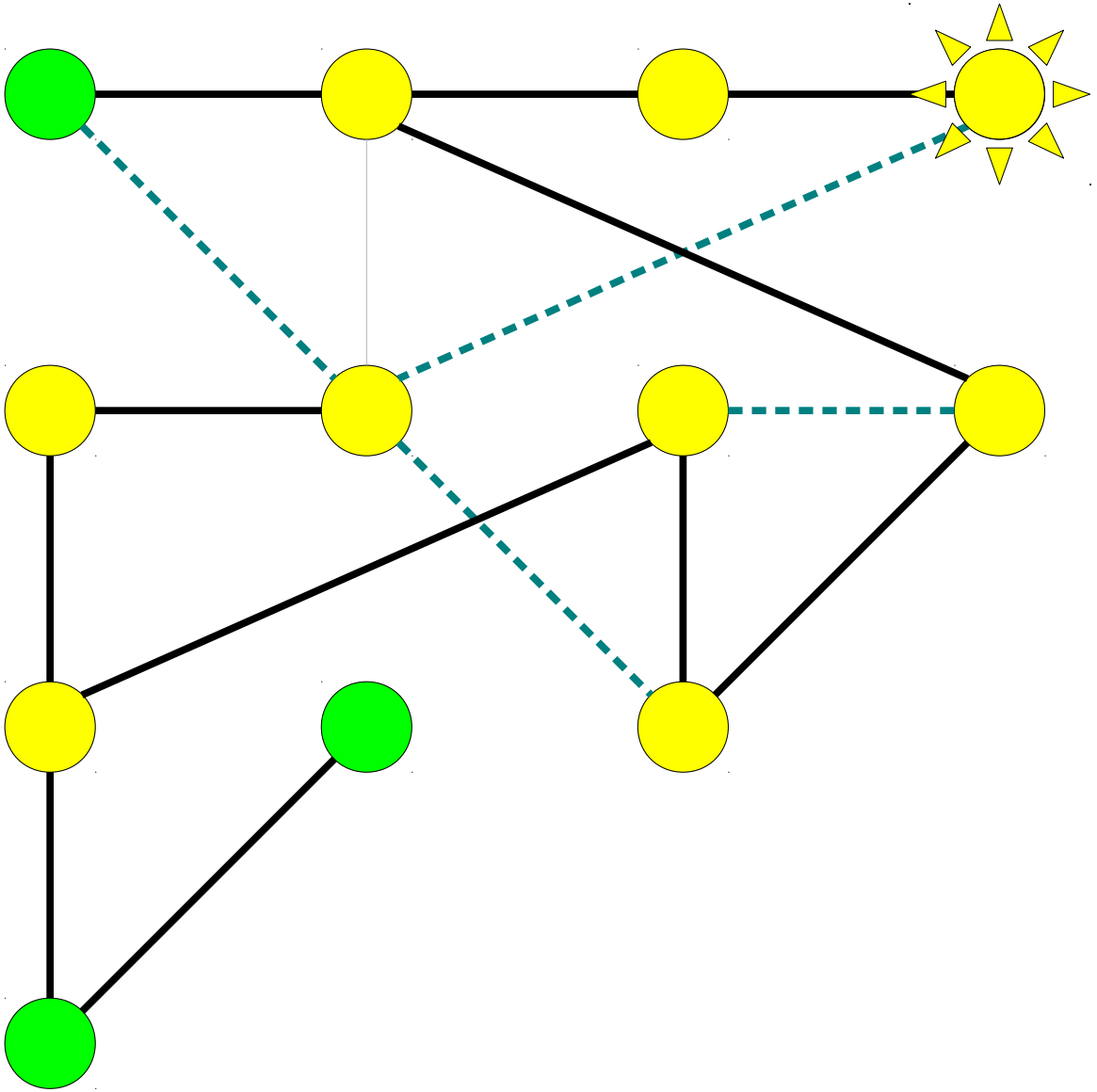
Depth-First Search



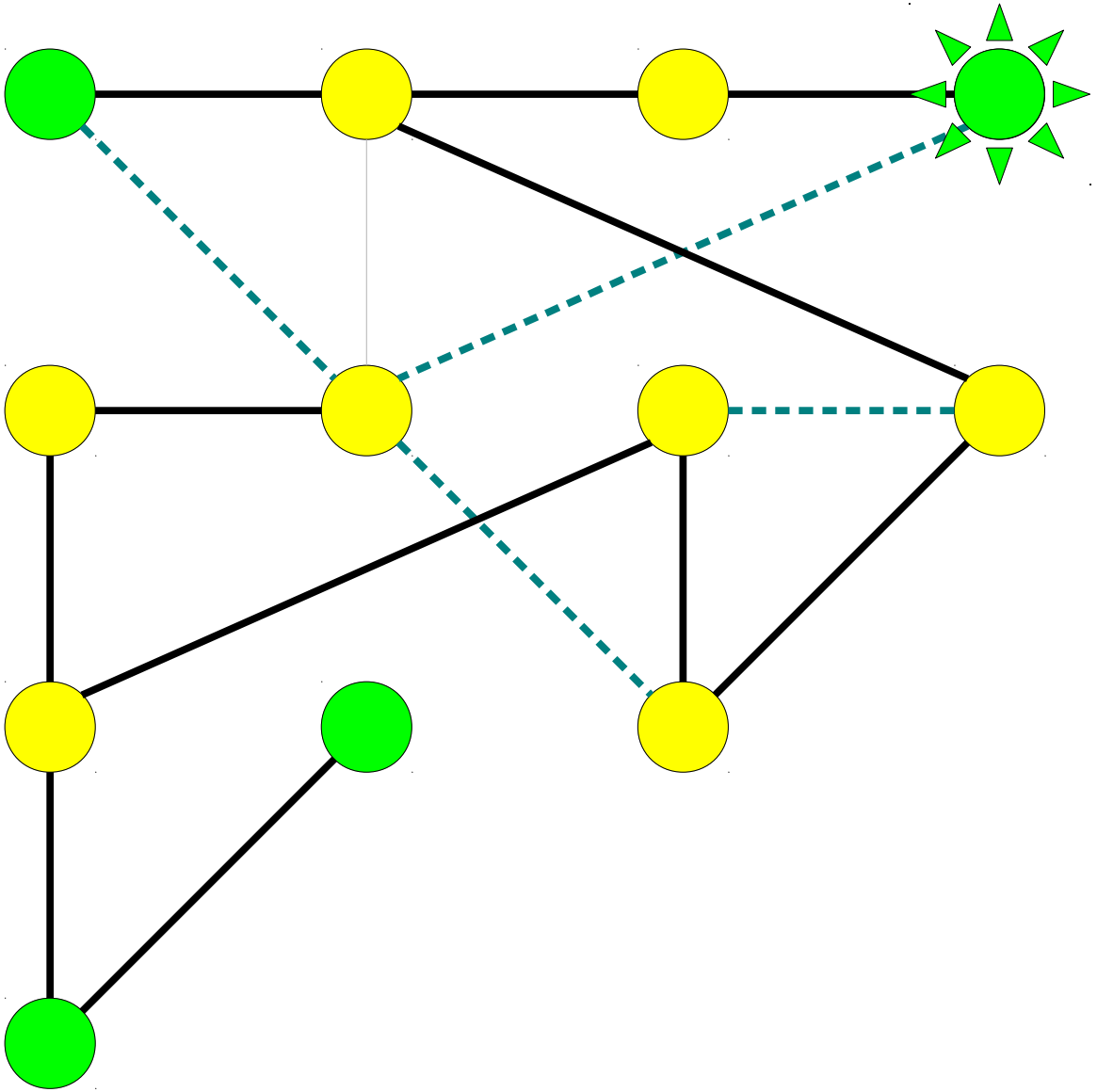
Depth-First Search



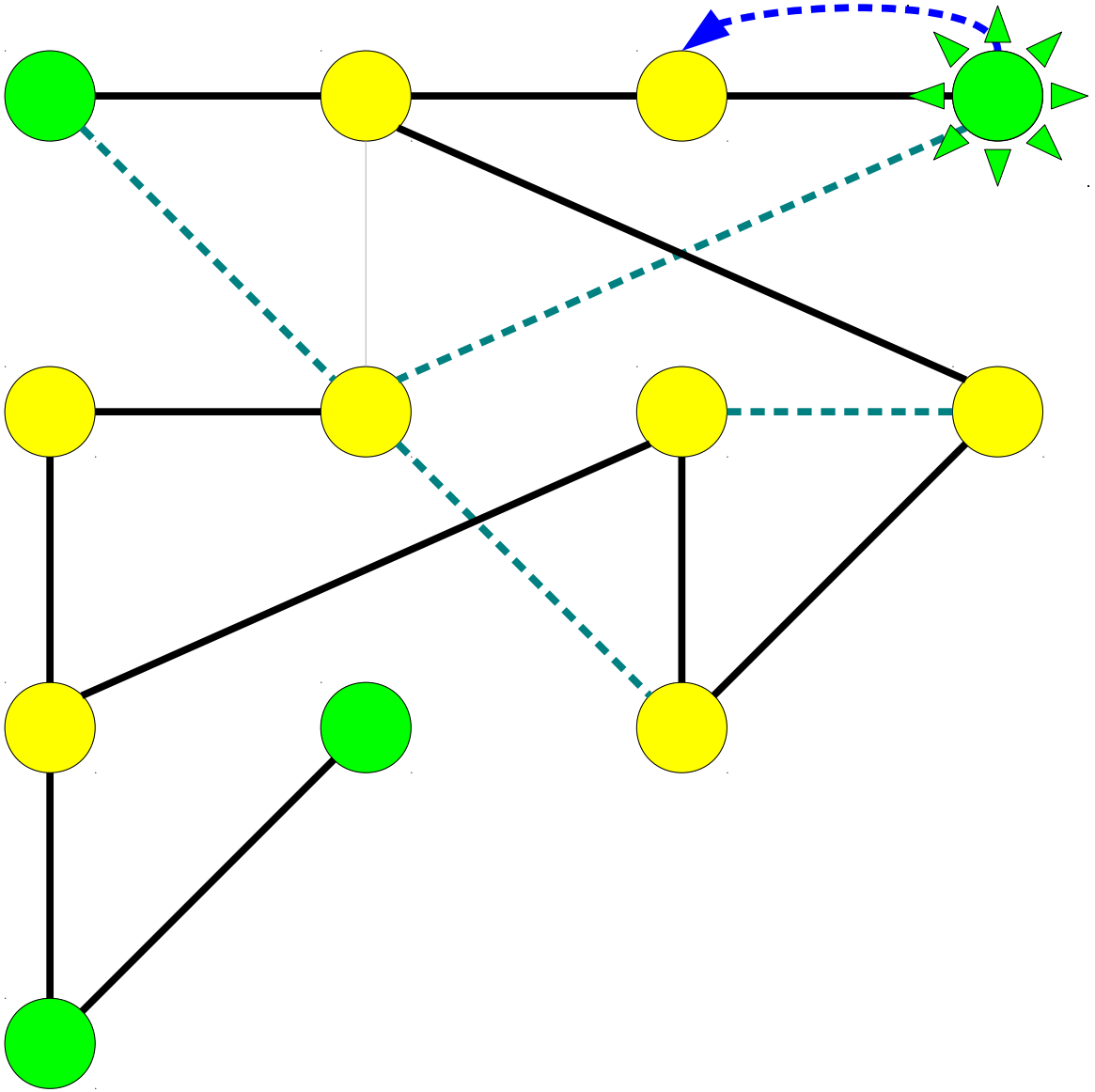
Depth-First Search



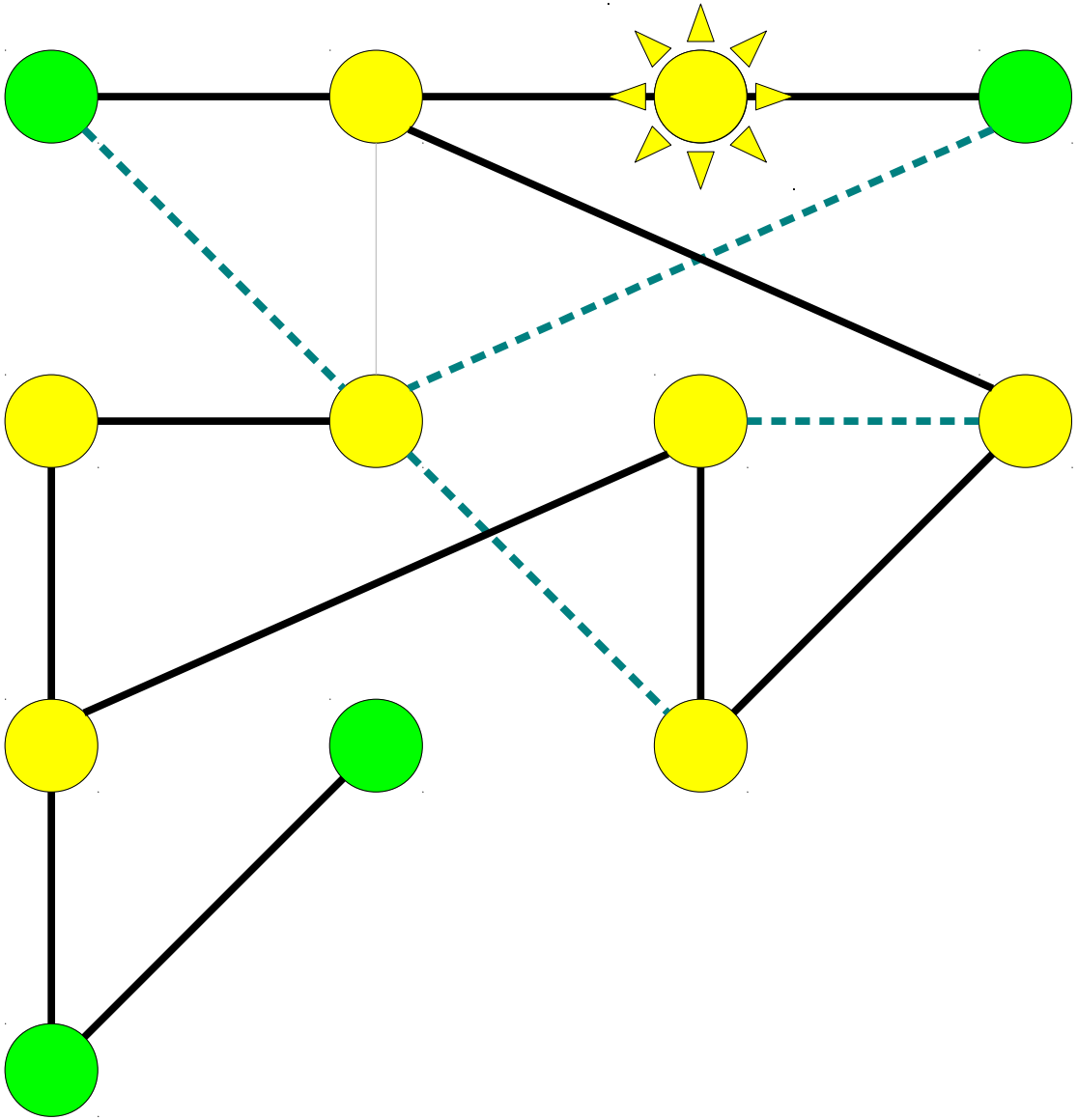
Depth-First Search



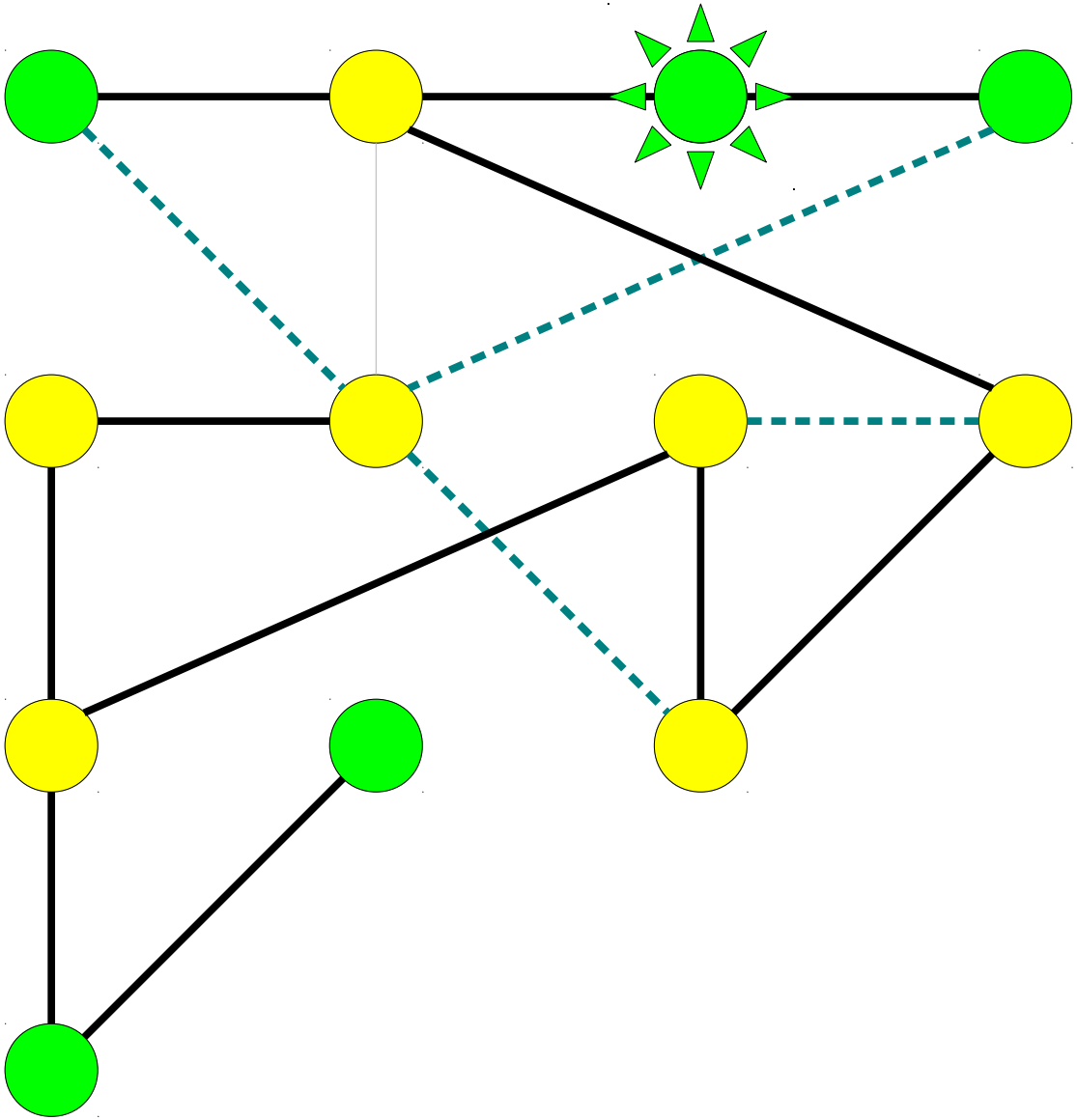
Depth-First Search



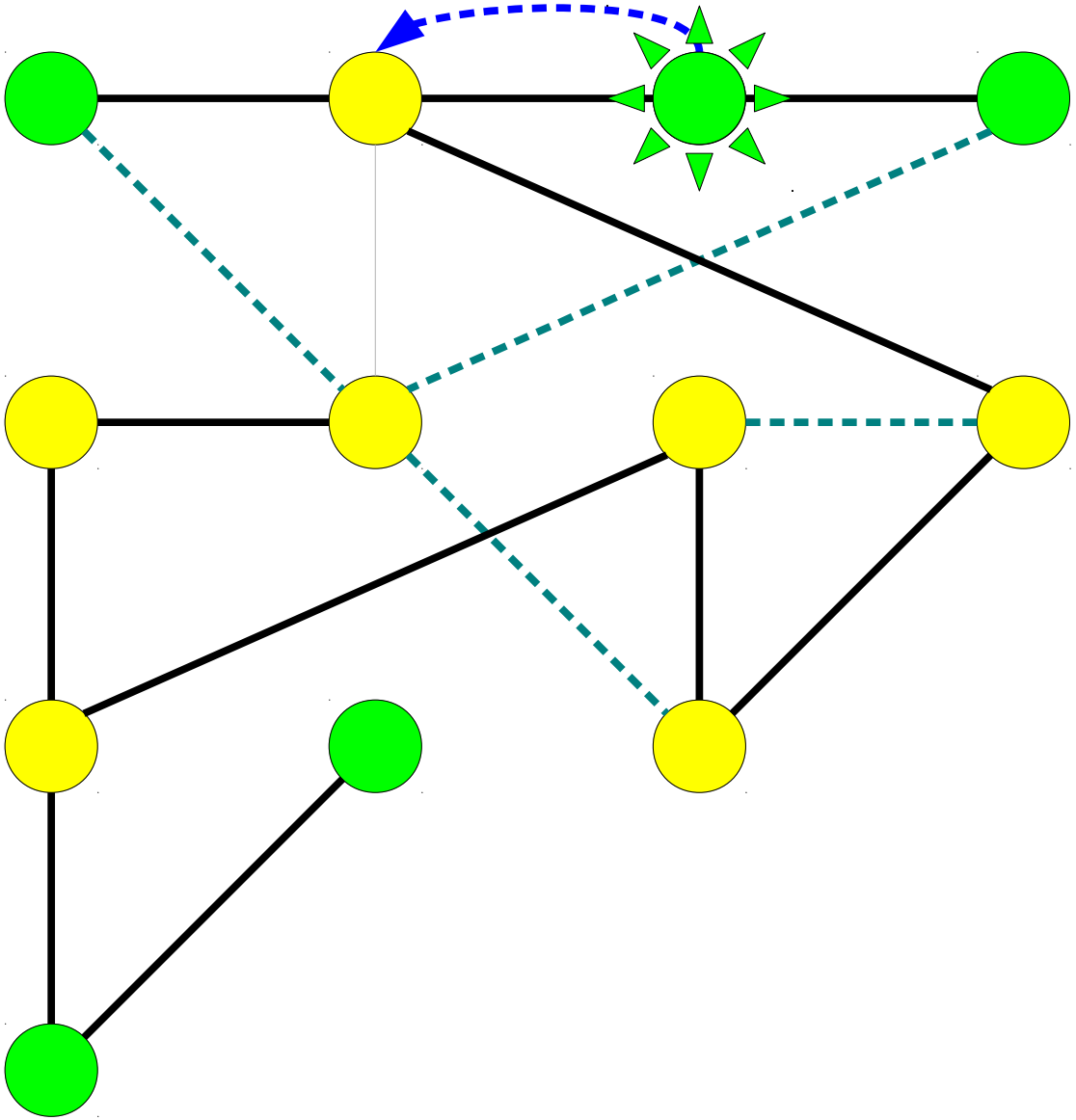
Depth-First Search



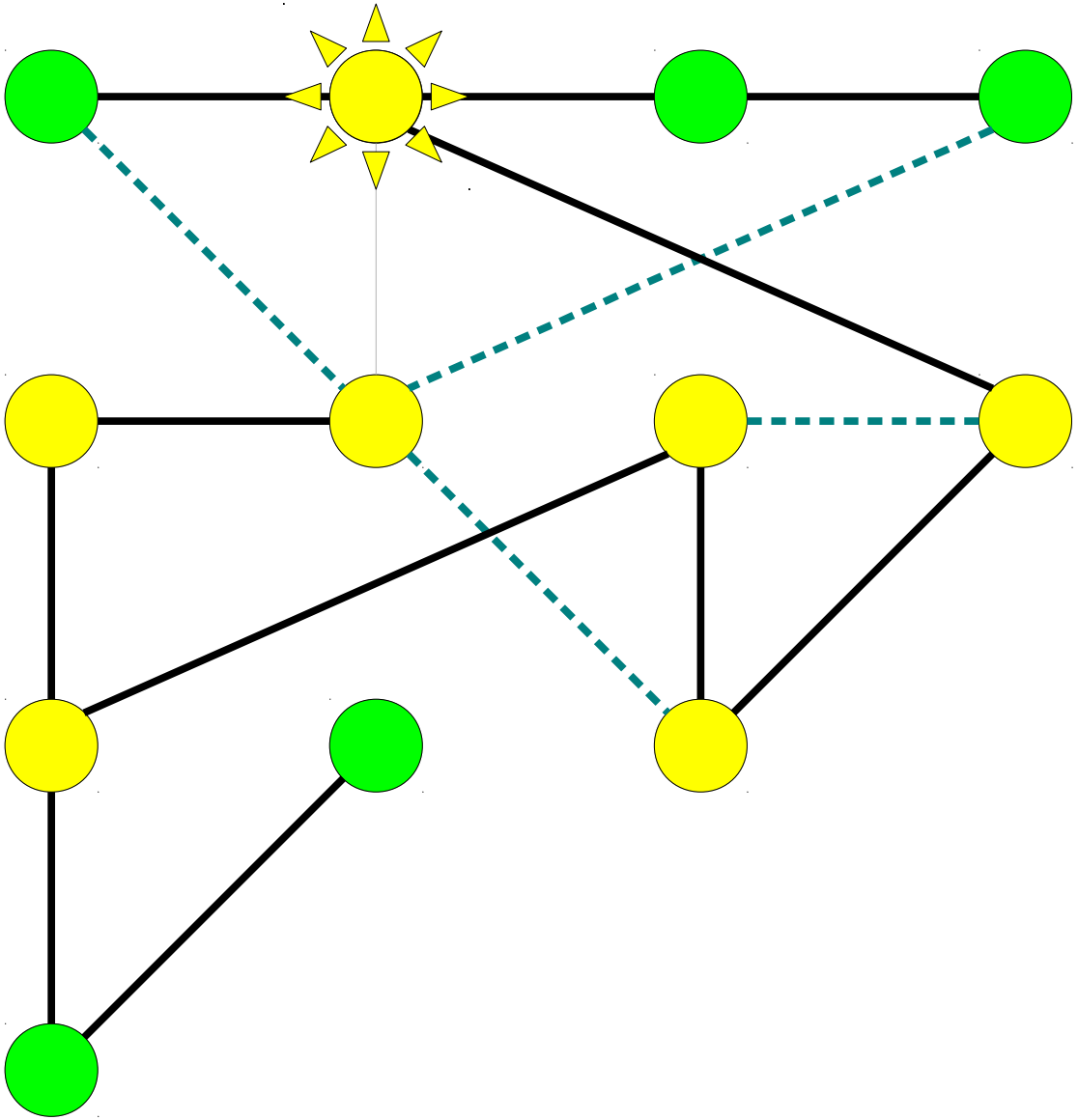
Depth-First Search



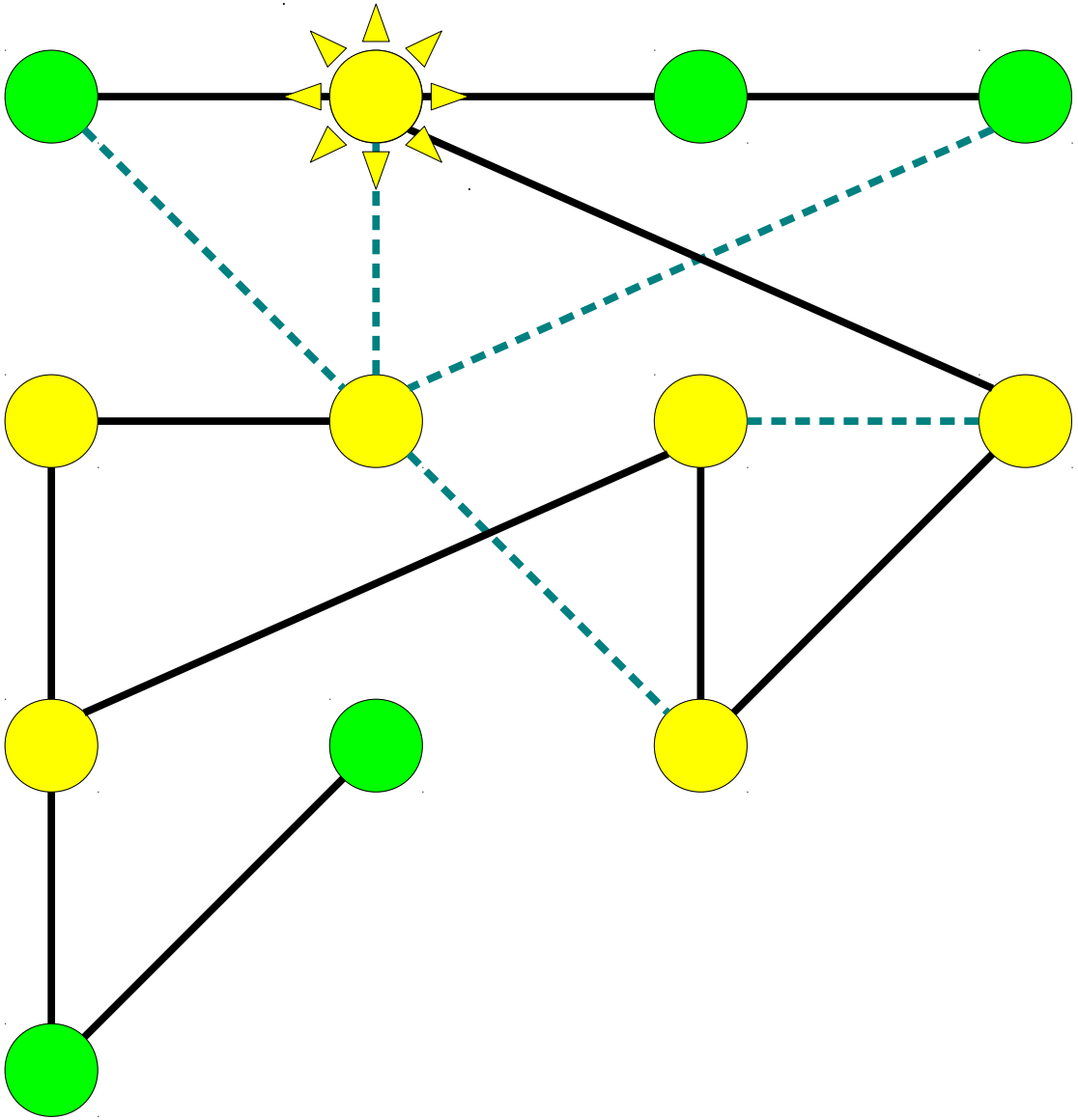
Depth-First Search



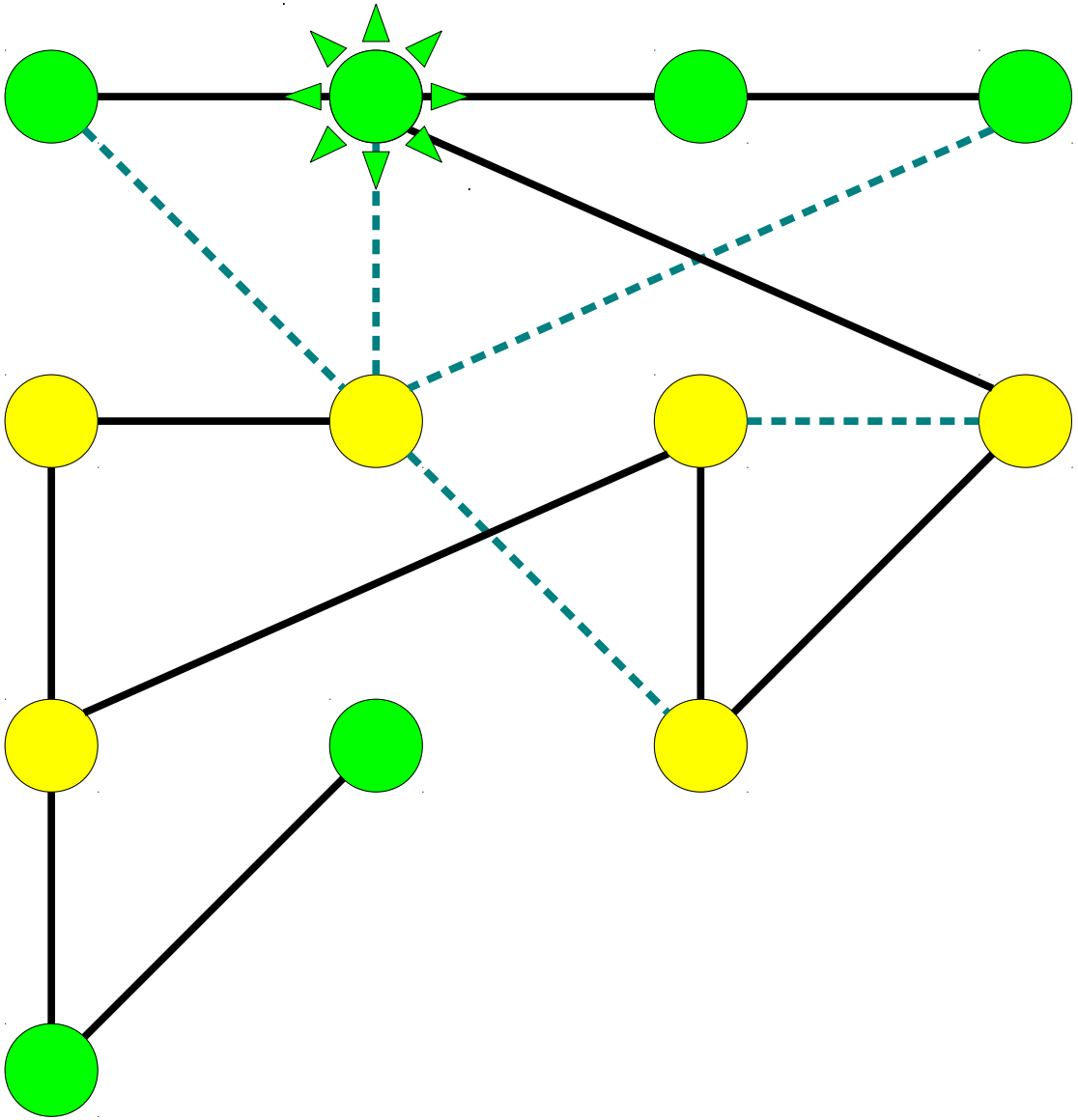
Depth-First Search



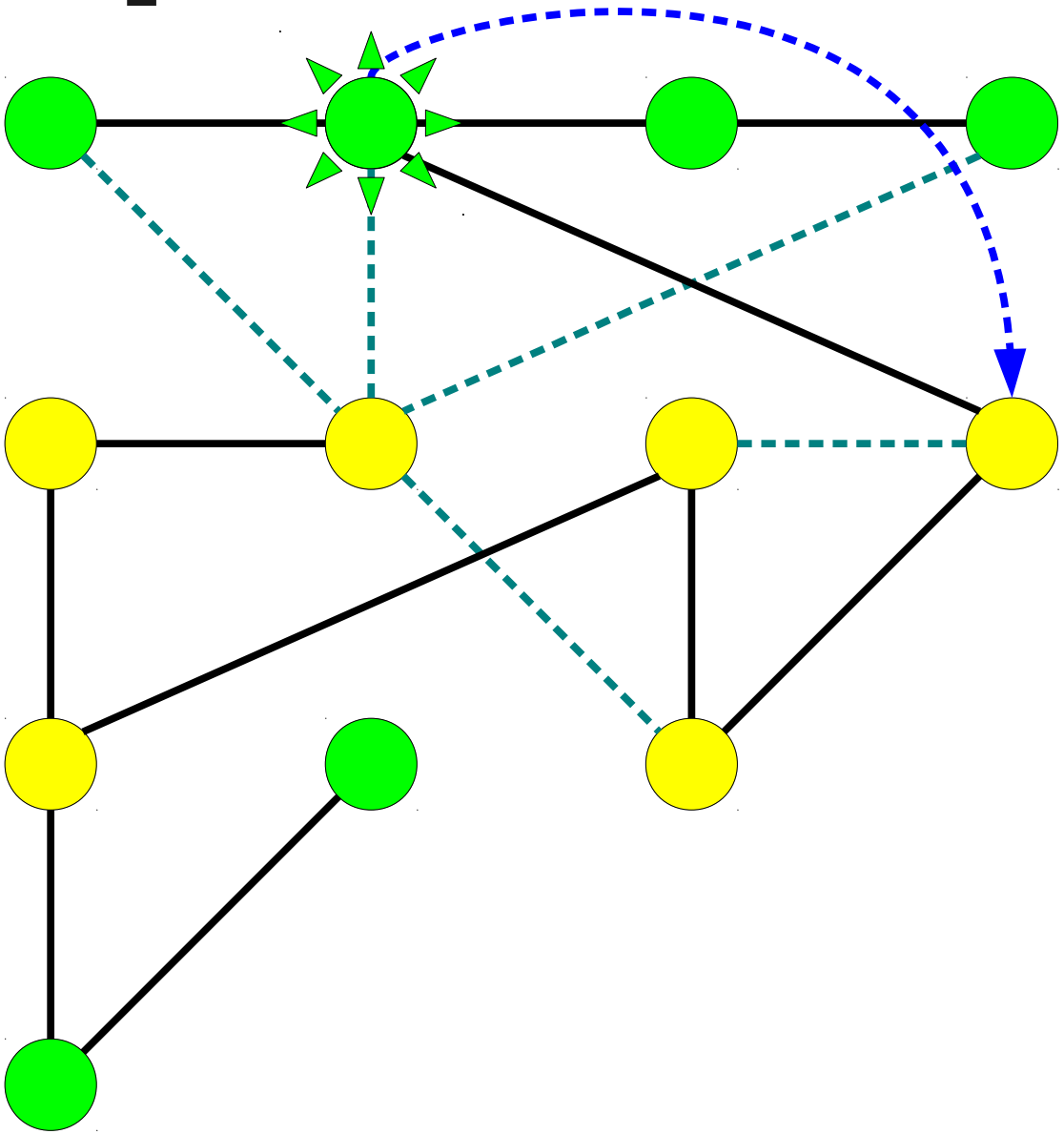
Depth-First Search



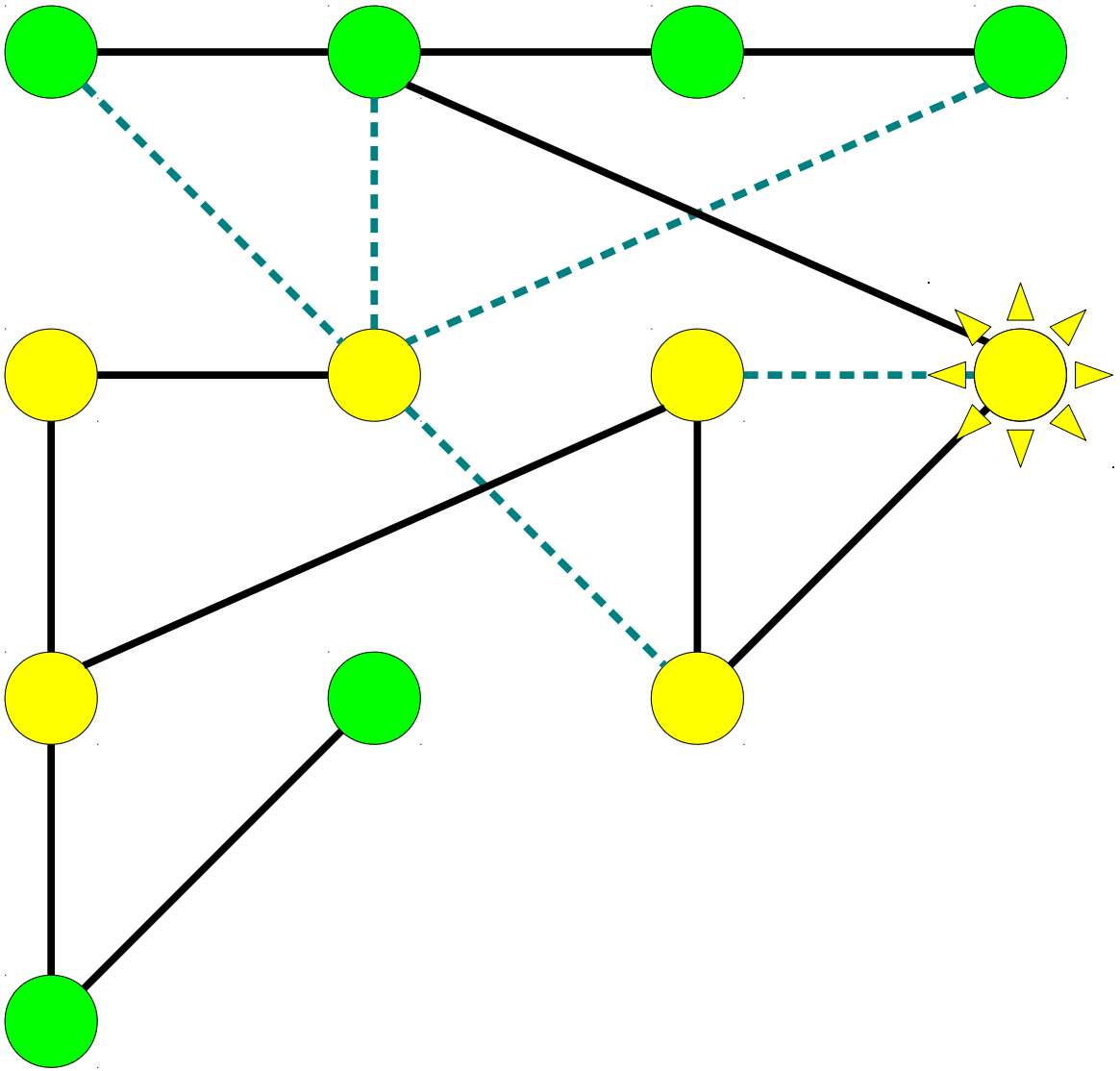
Depth-First Search



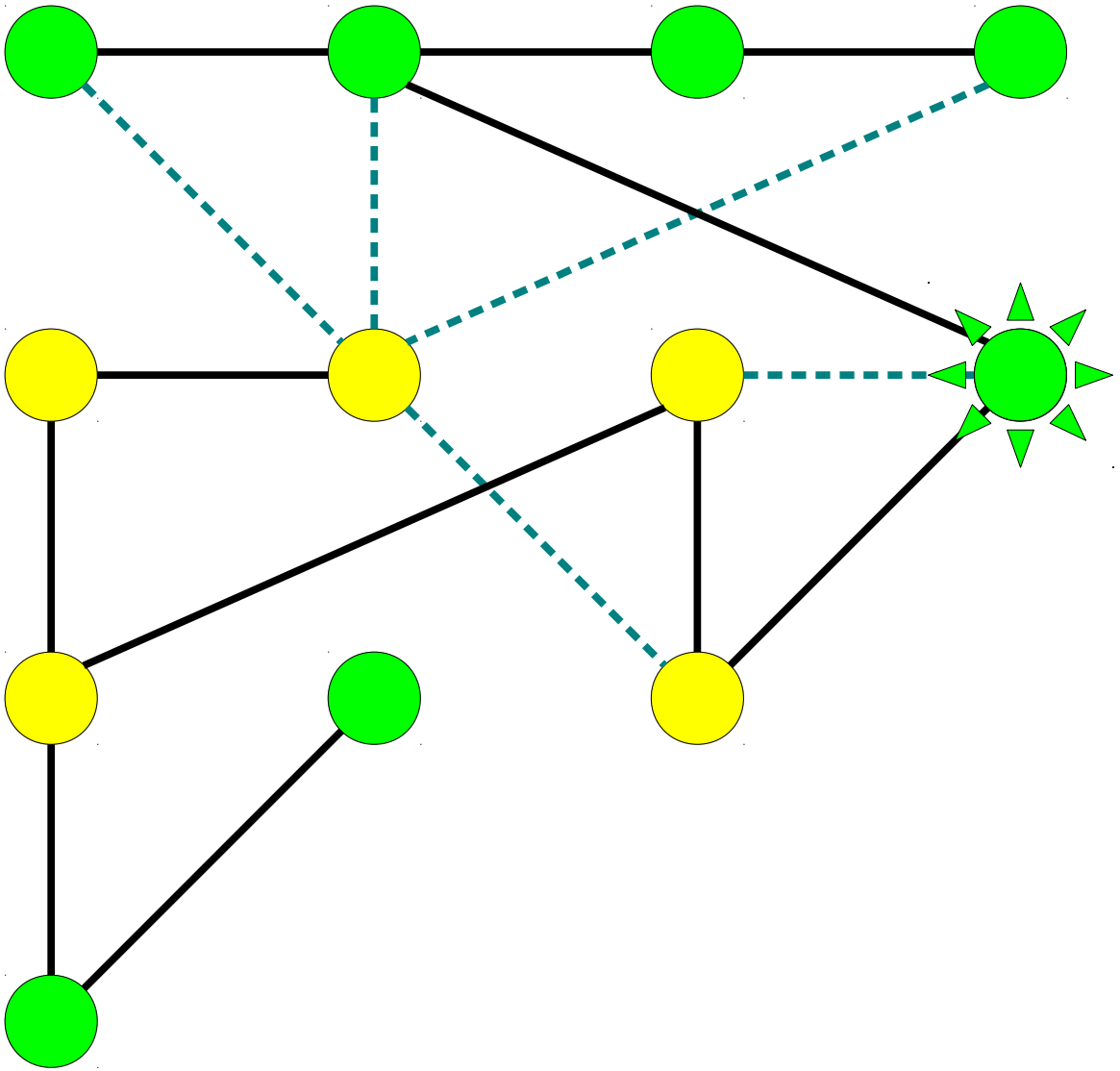
Depth-First Search



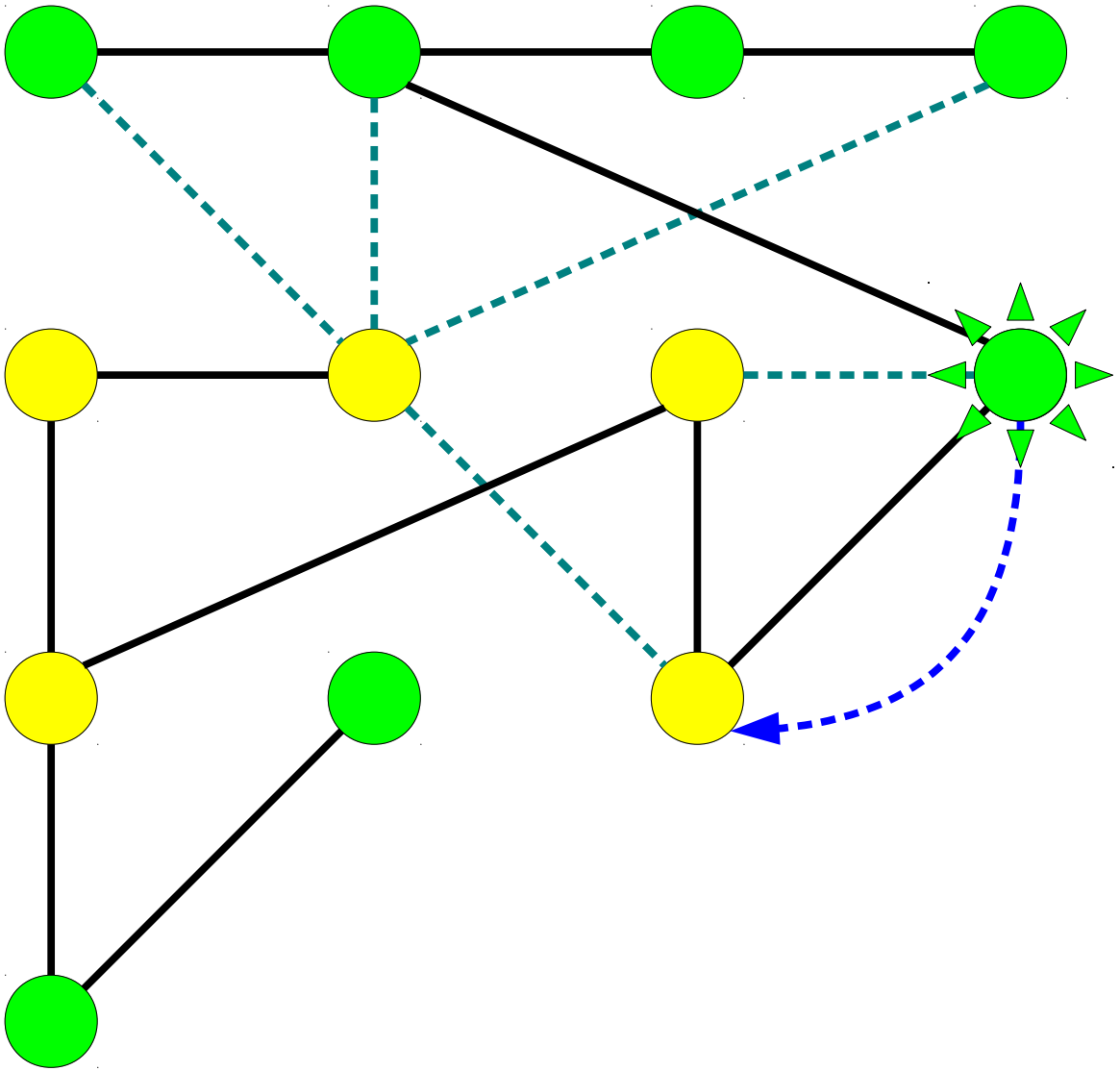
Depth-First Search



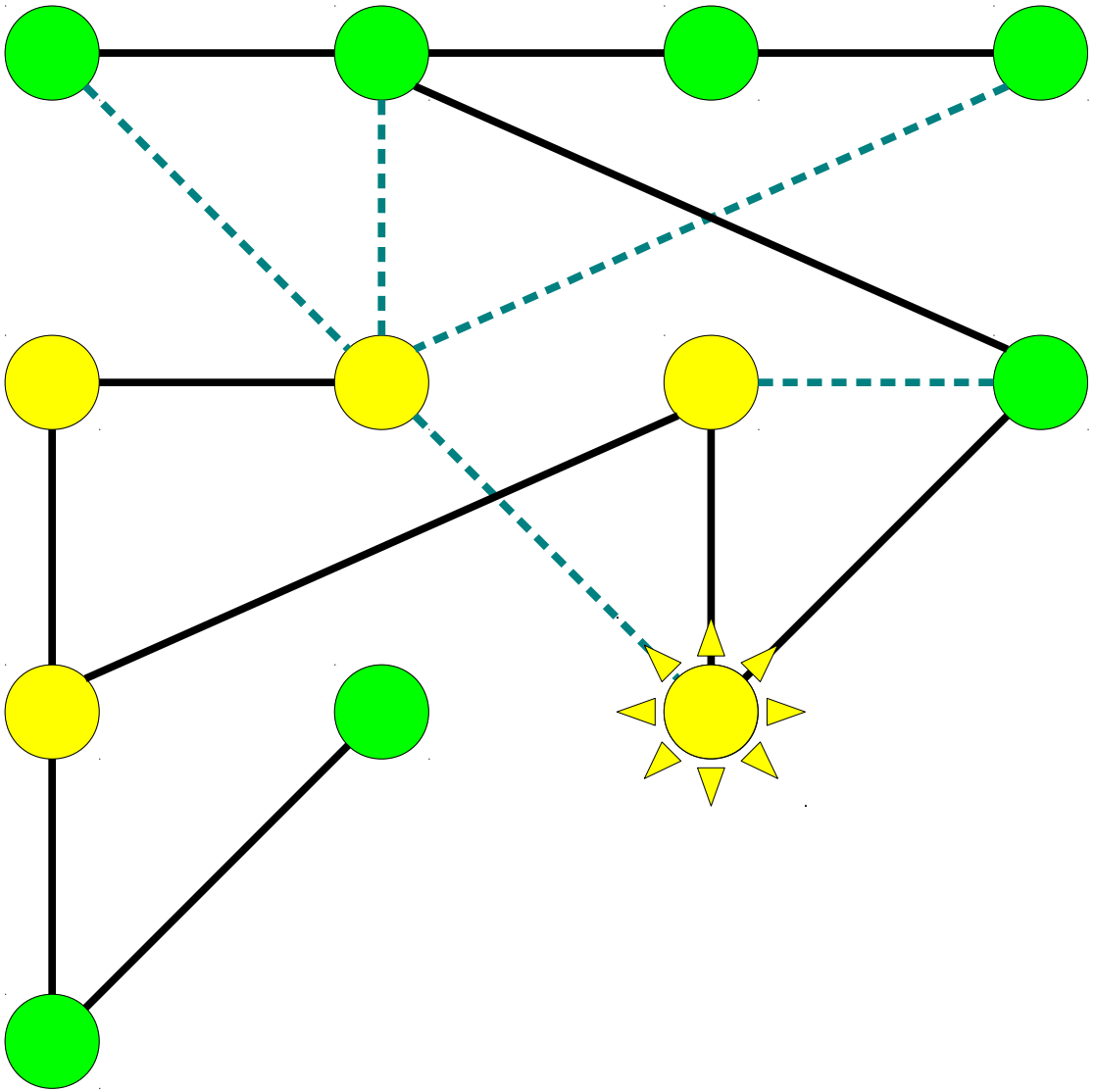
Depth-First Search



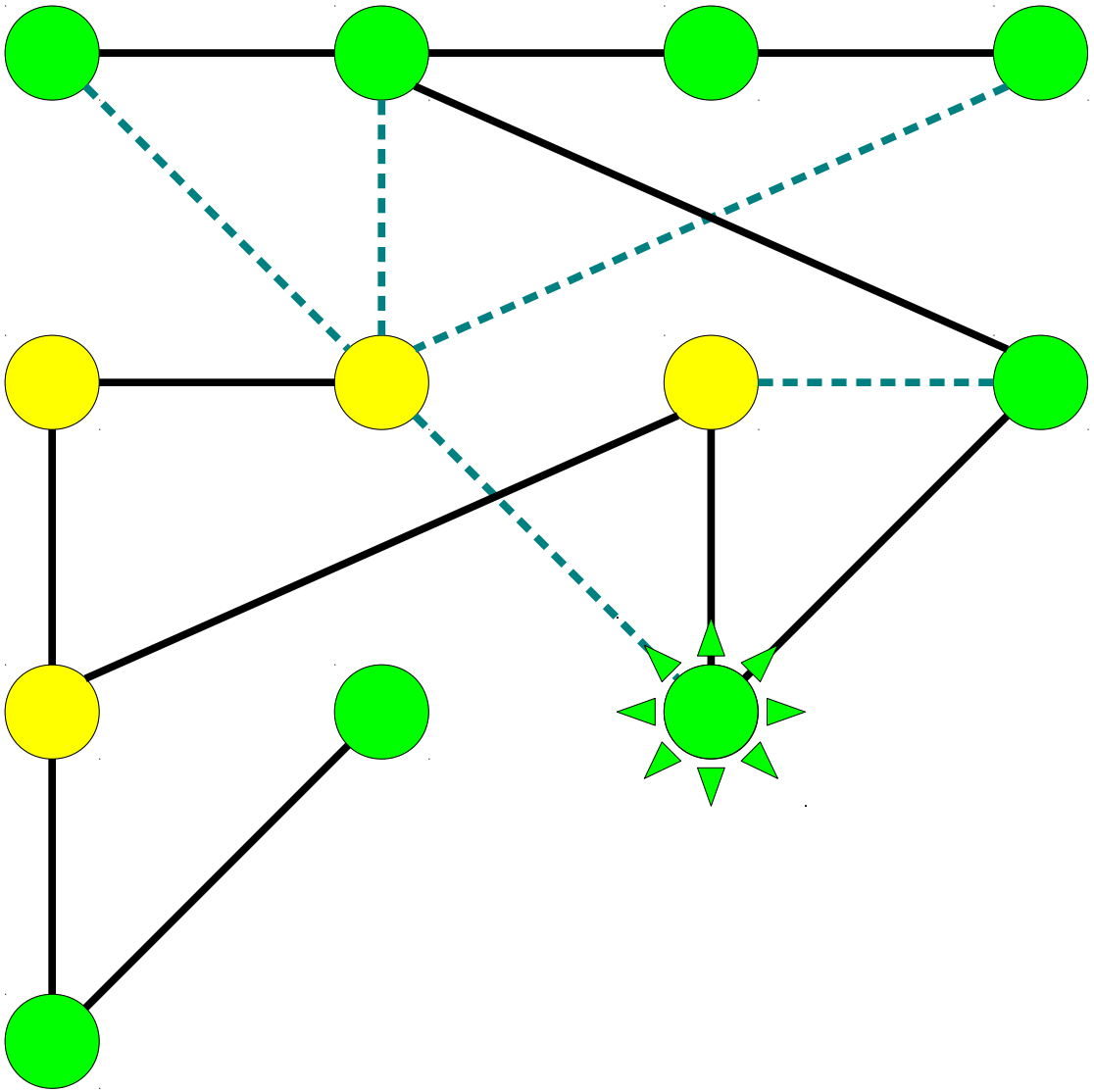
Depth-First Search



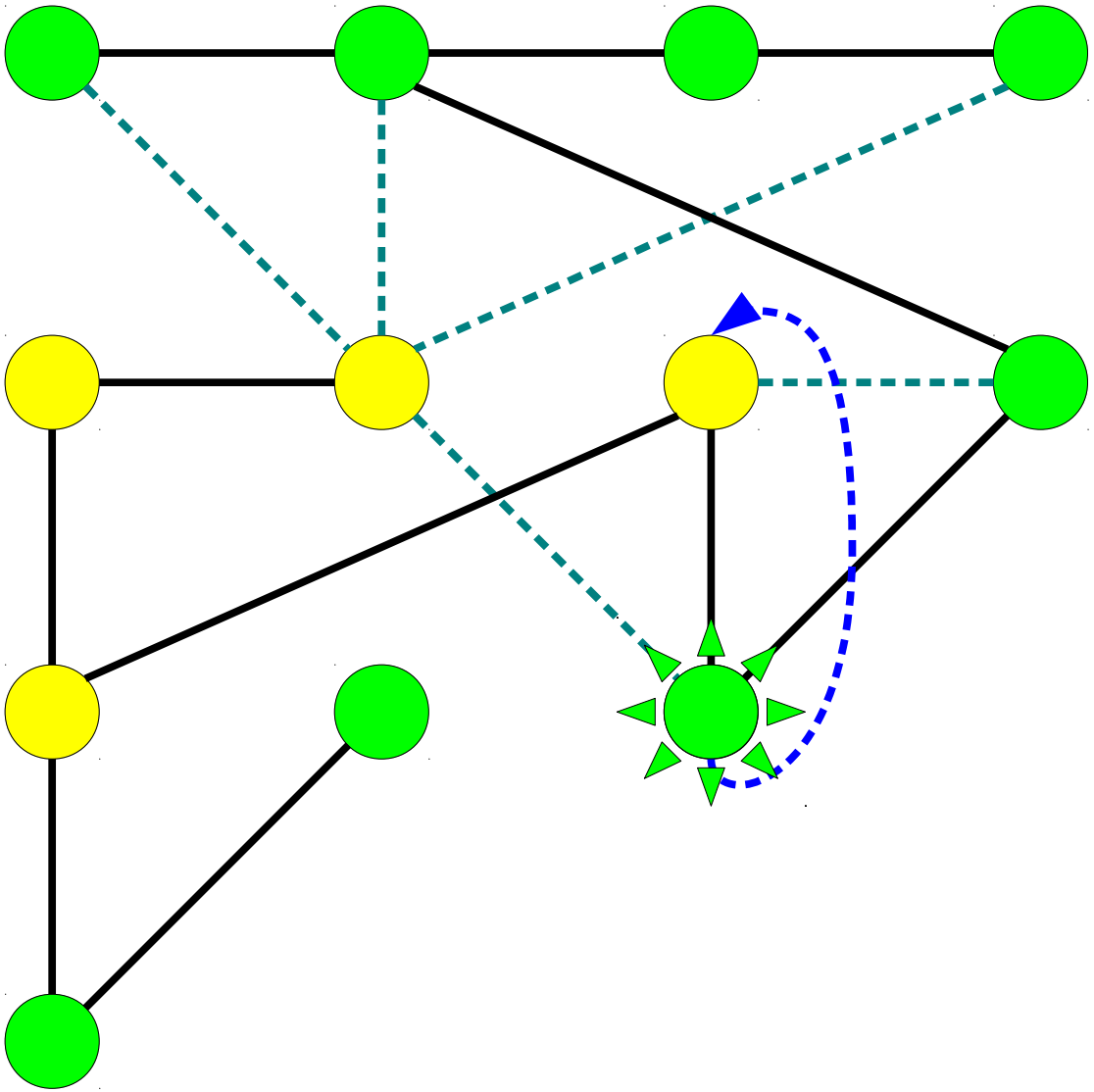
Depth-First Search



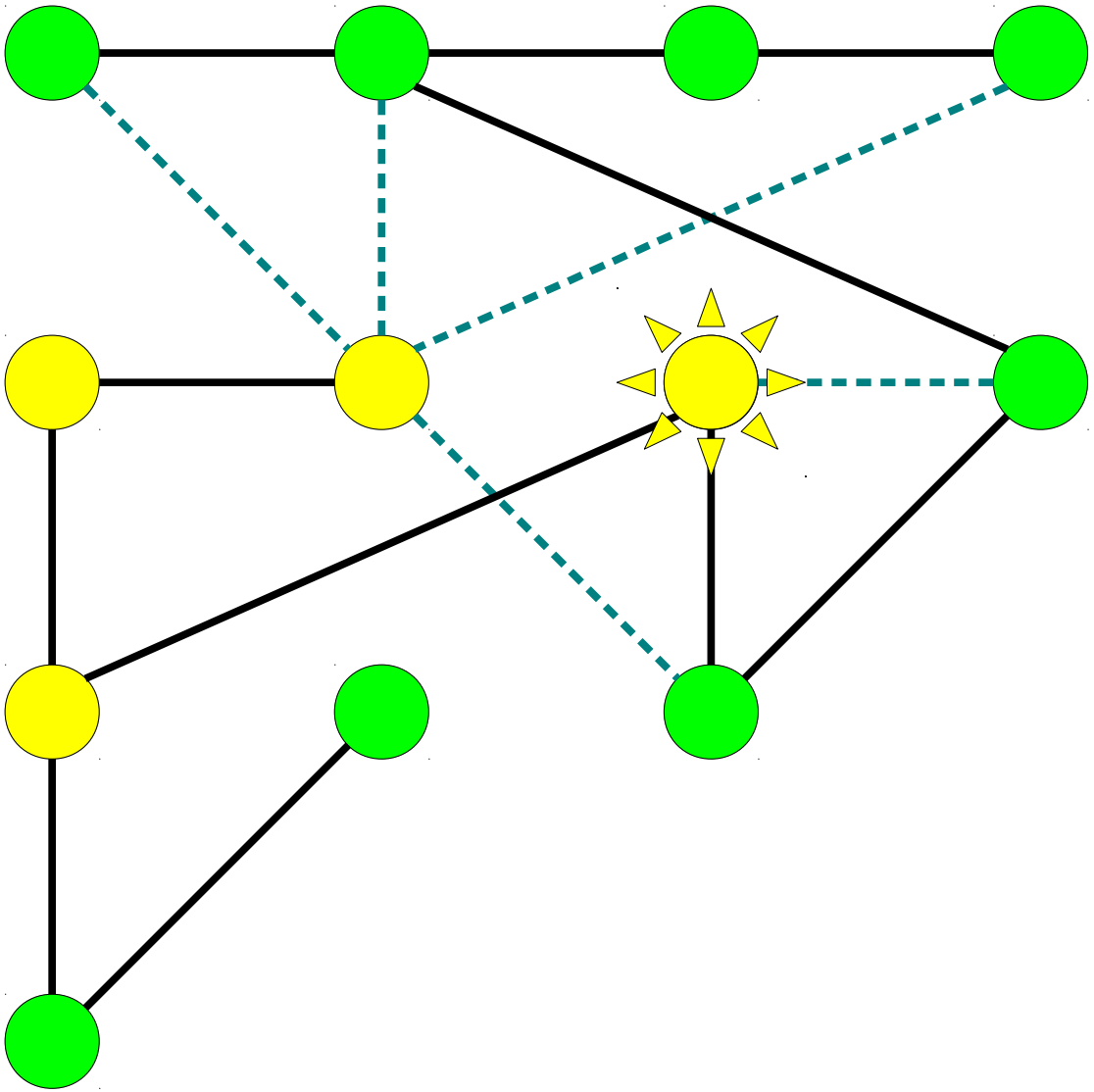
Depth-First Search



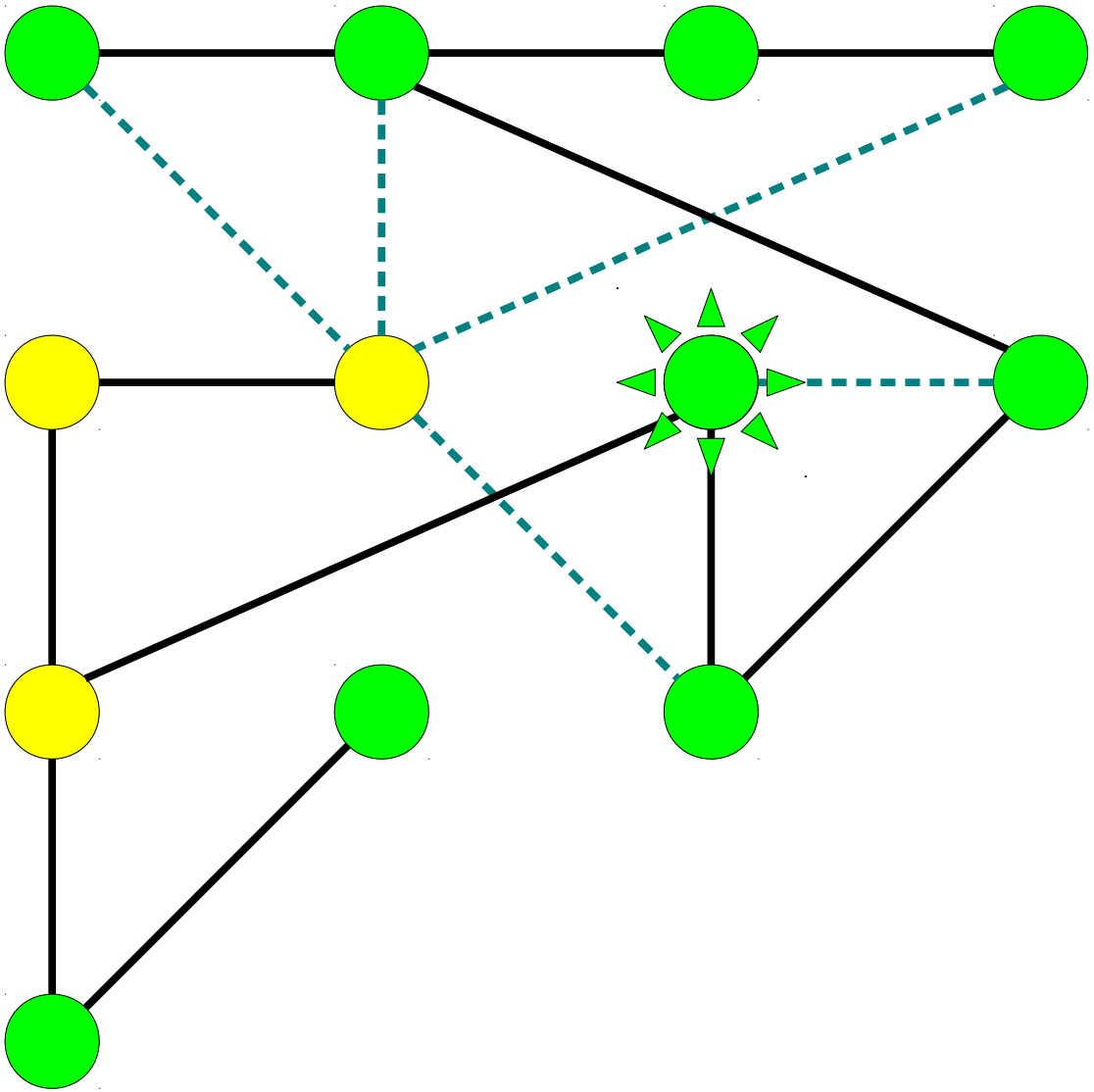
Depth-First Search



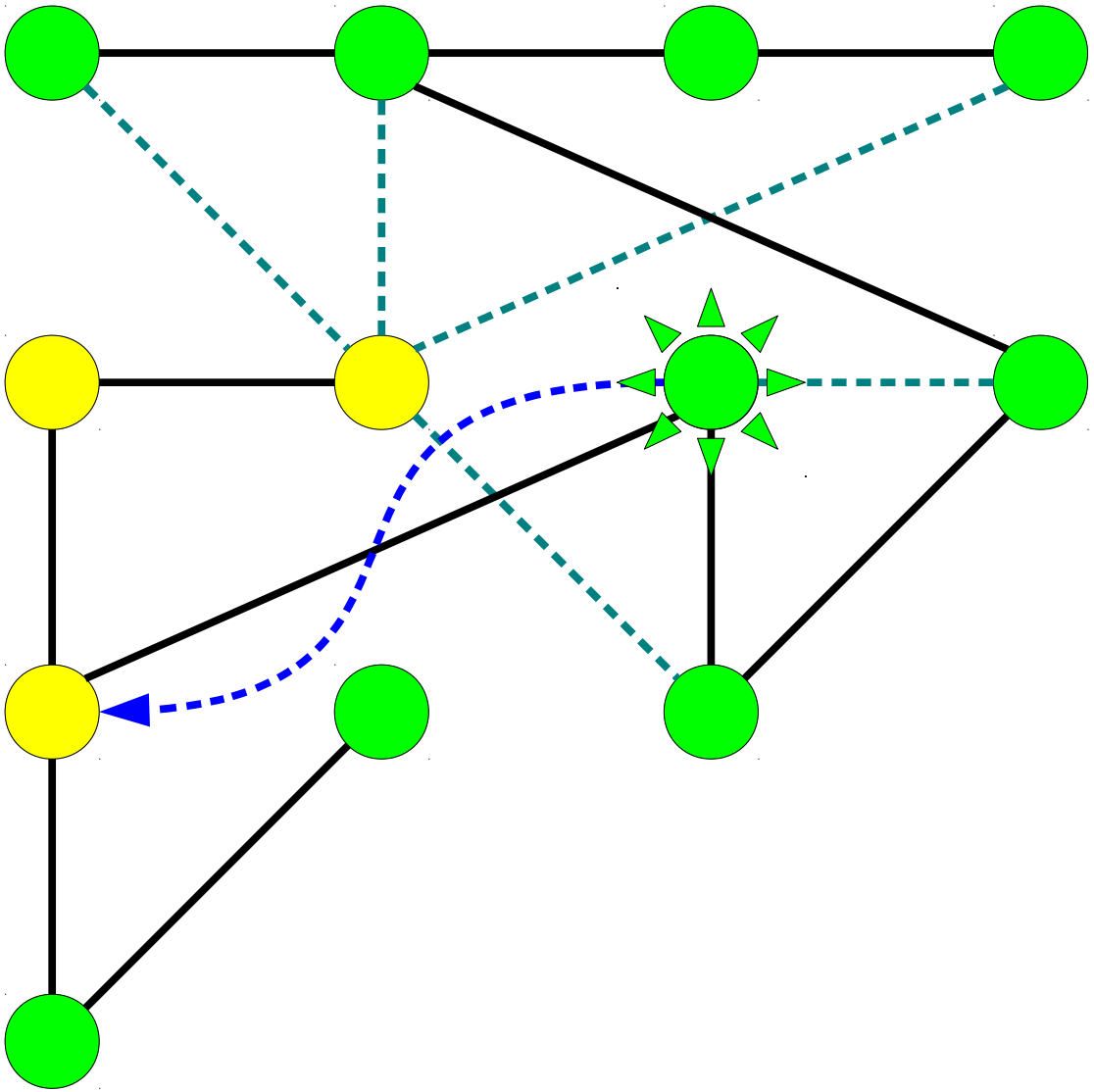
Depth-First Search



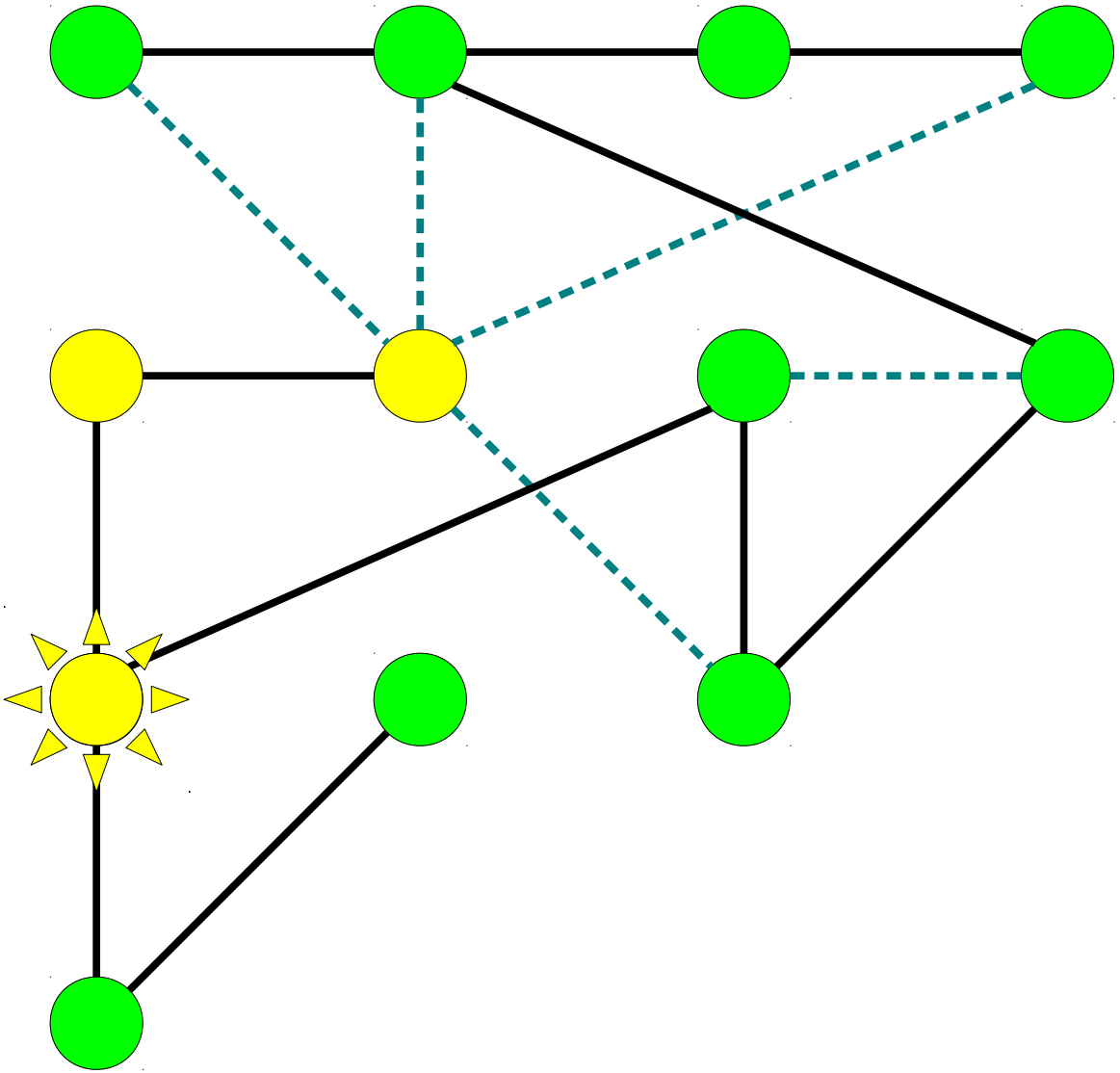
Depth-First Search



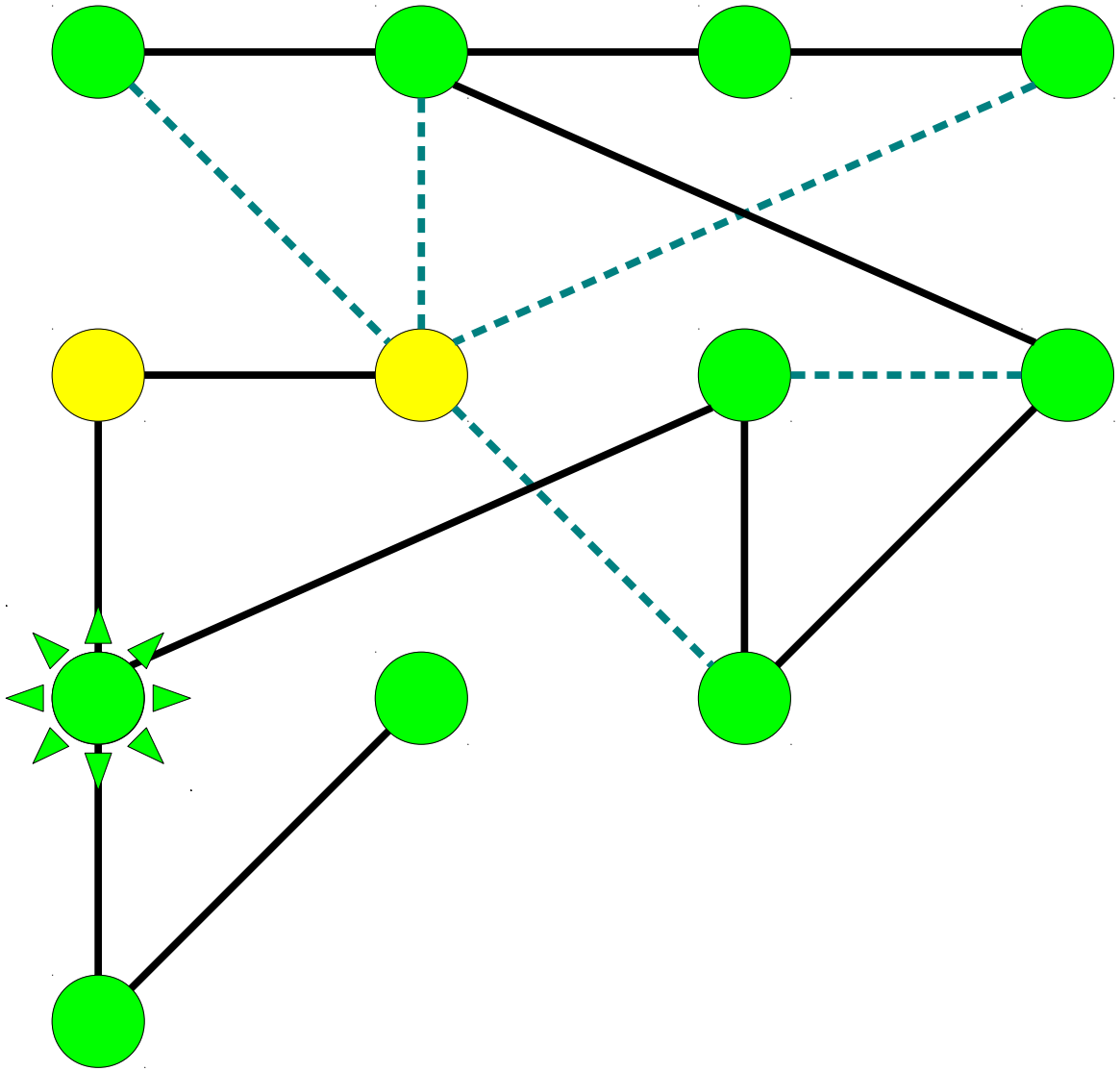
Depth-First Search



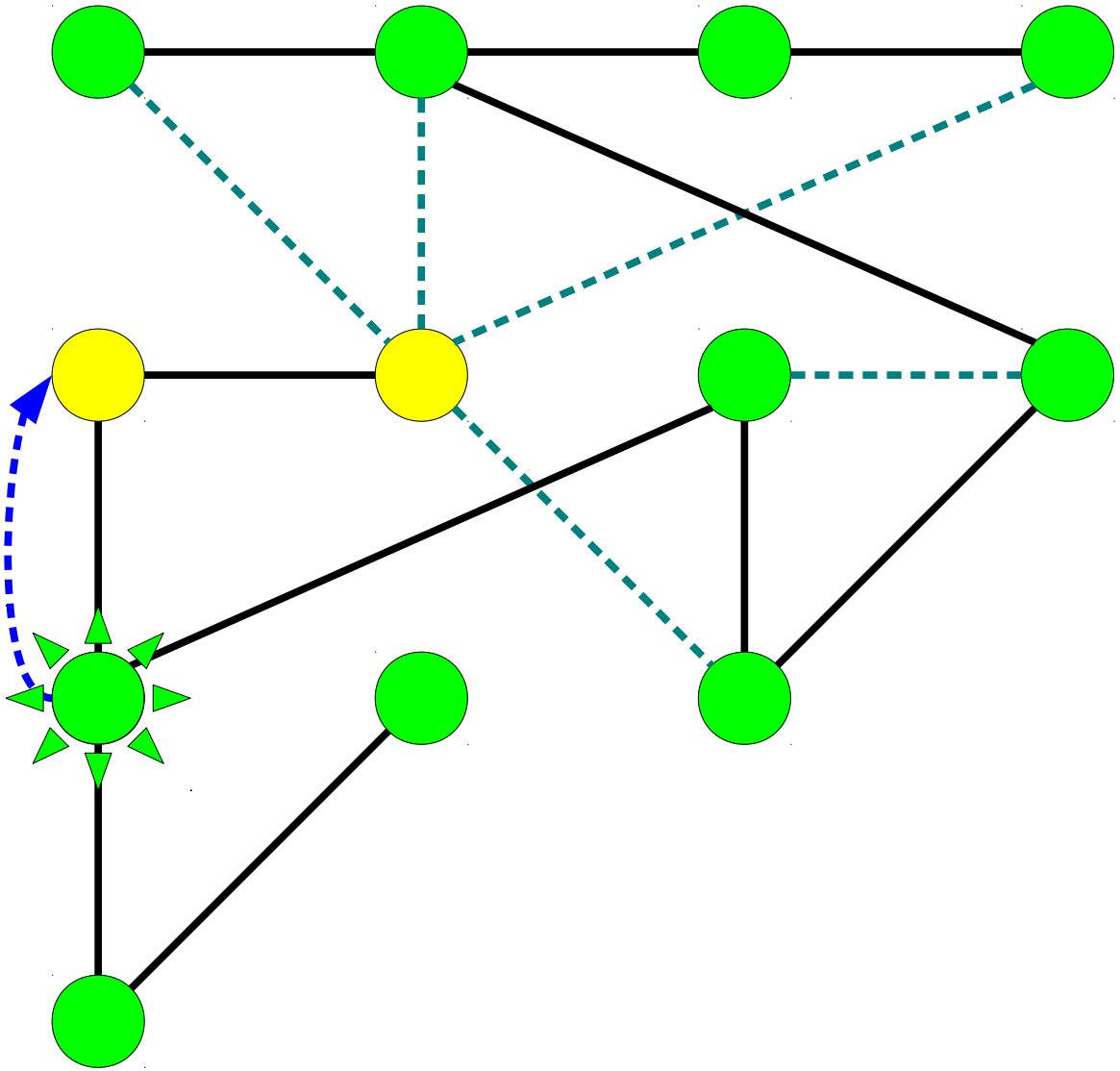
Depth-First Search



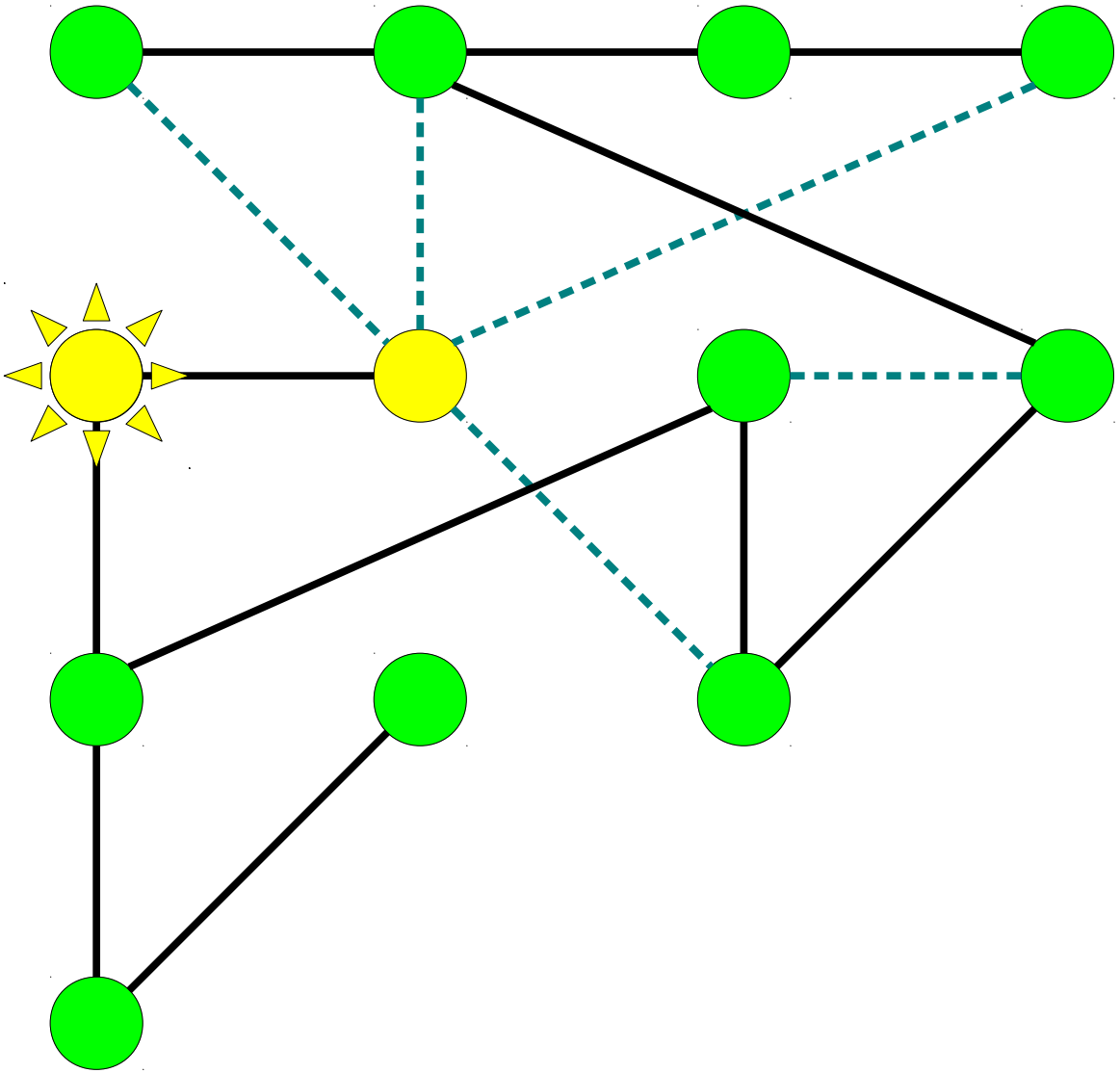
Depth-First Search



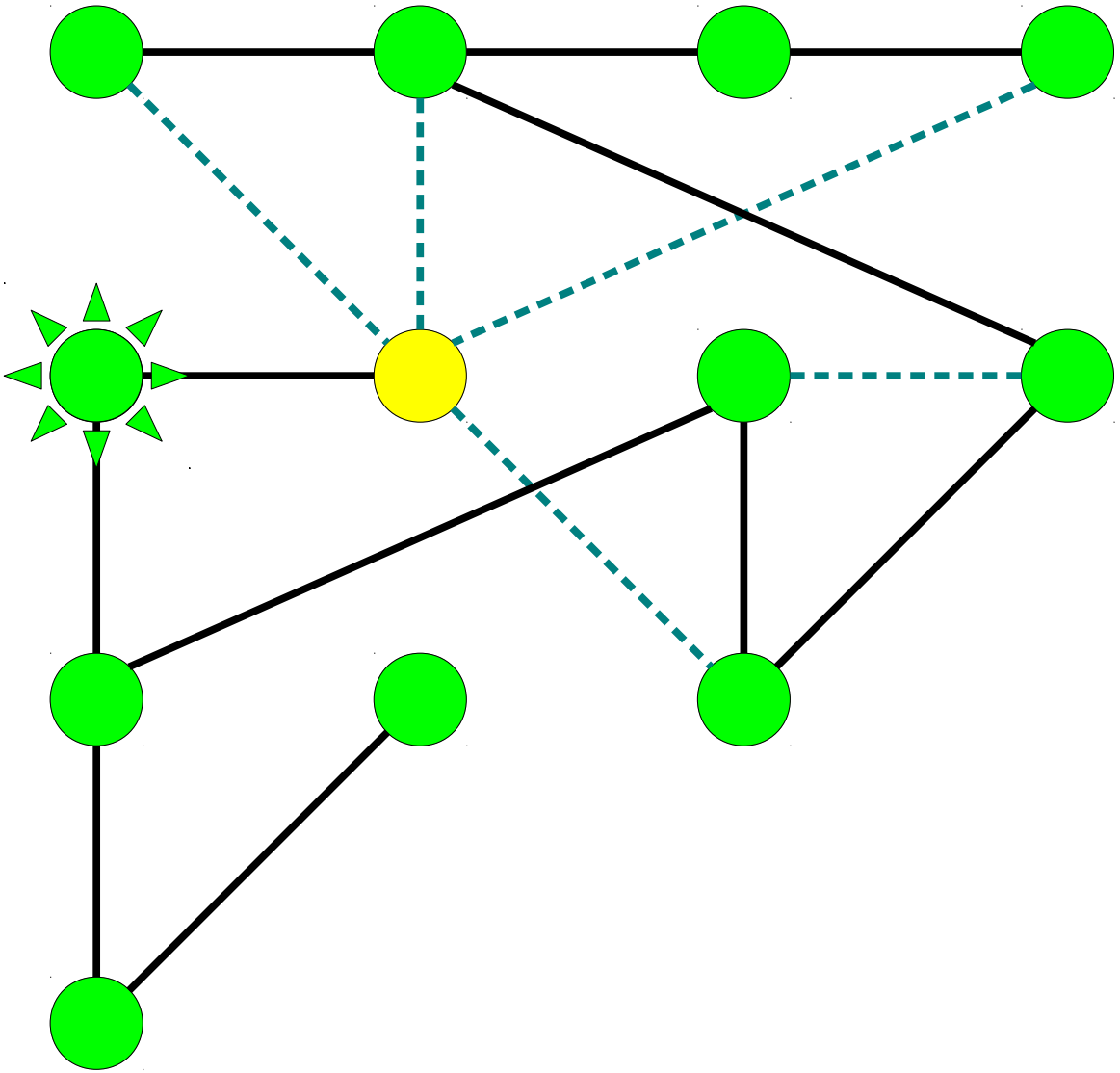
Depth-First Search



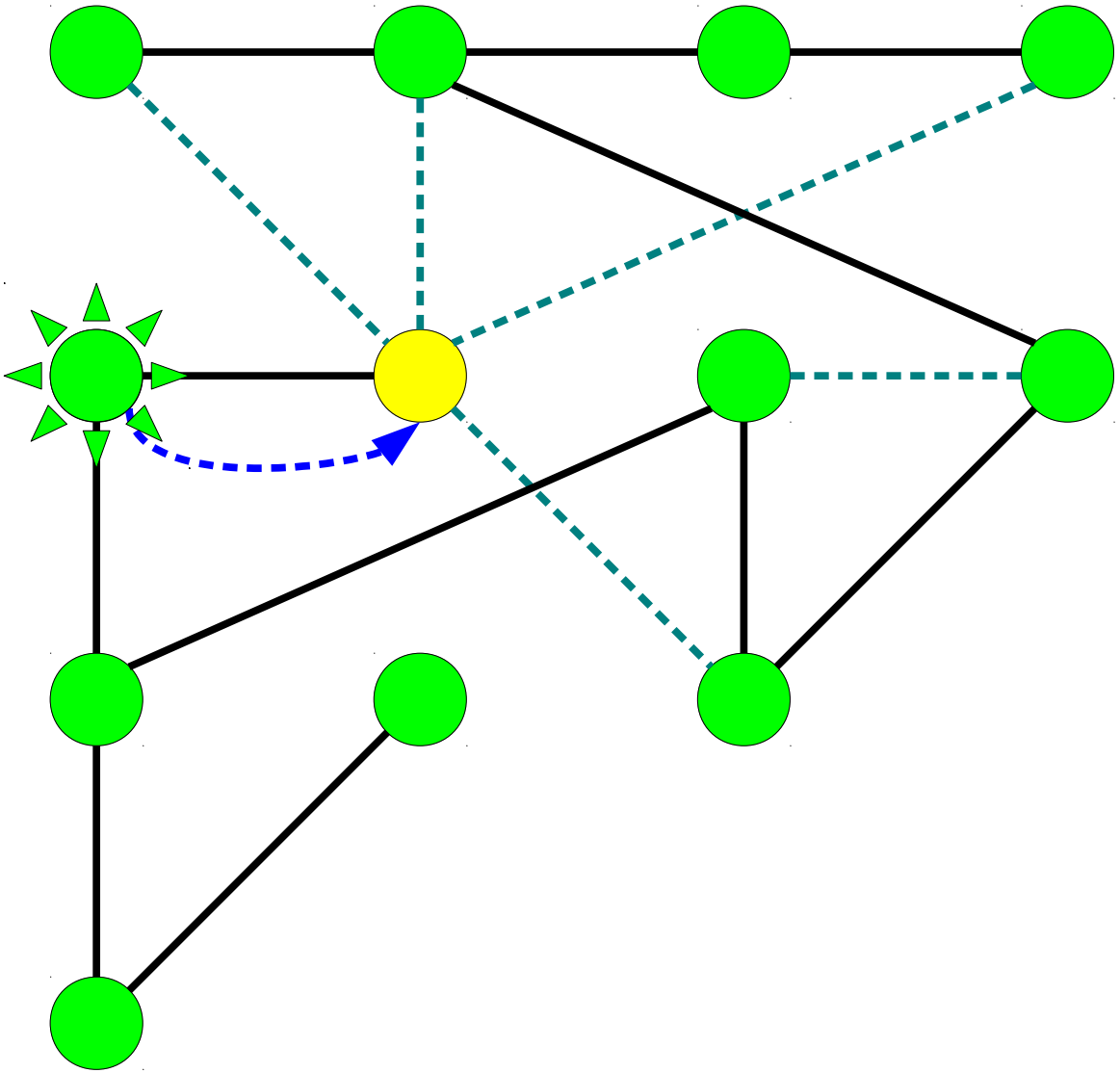
Depth-First Search



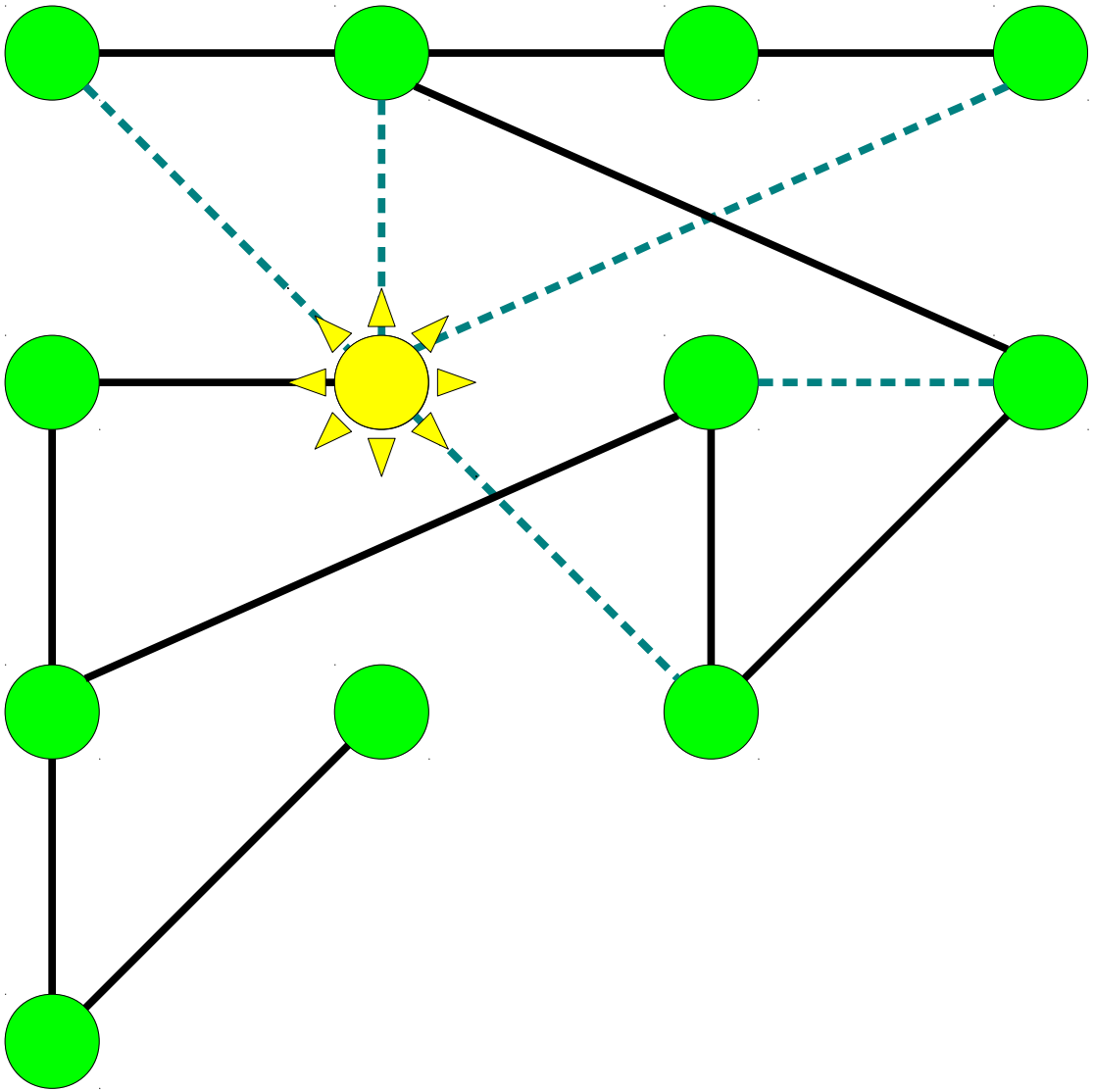
Depth-First Search



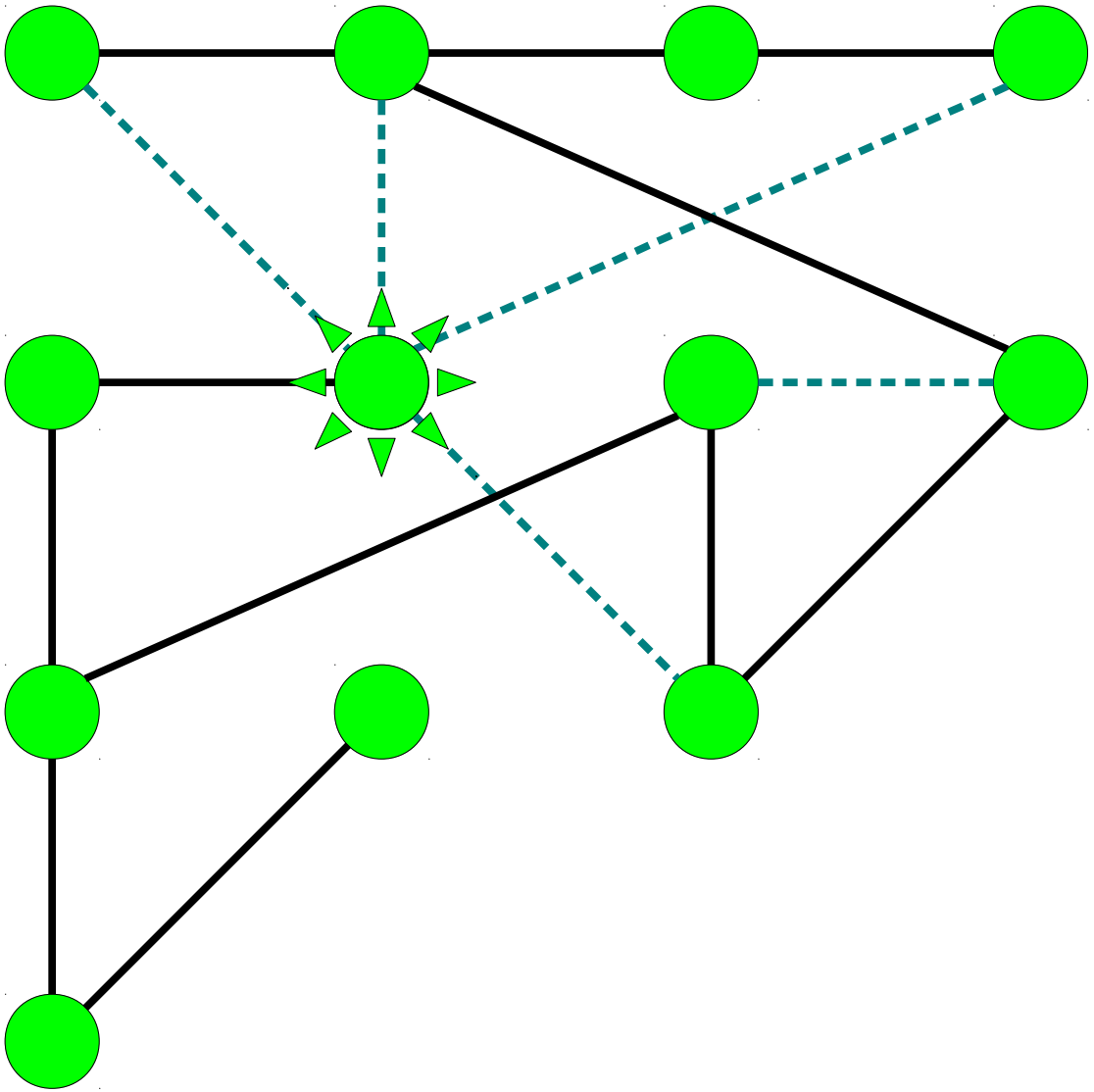
Depth-First Search



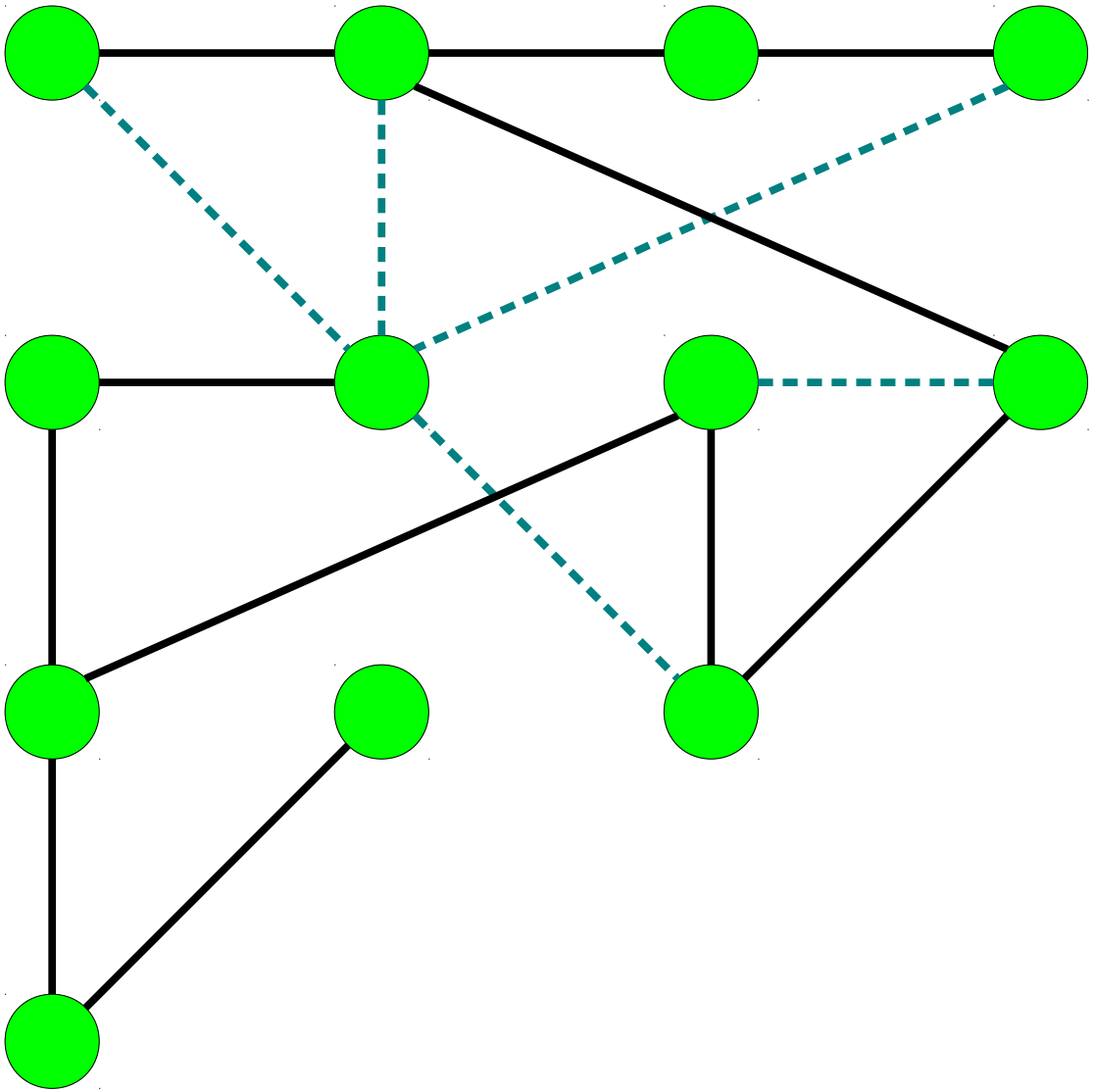
Depth-First Search



Depth-First Search



Depth-First Search

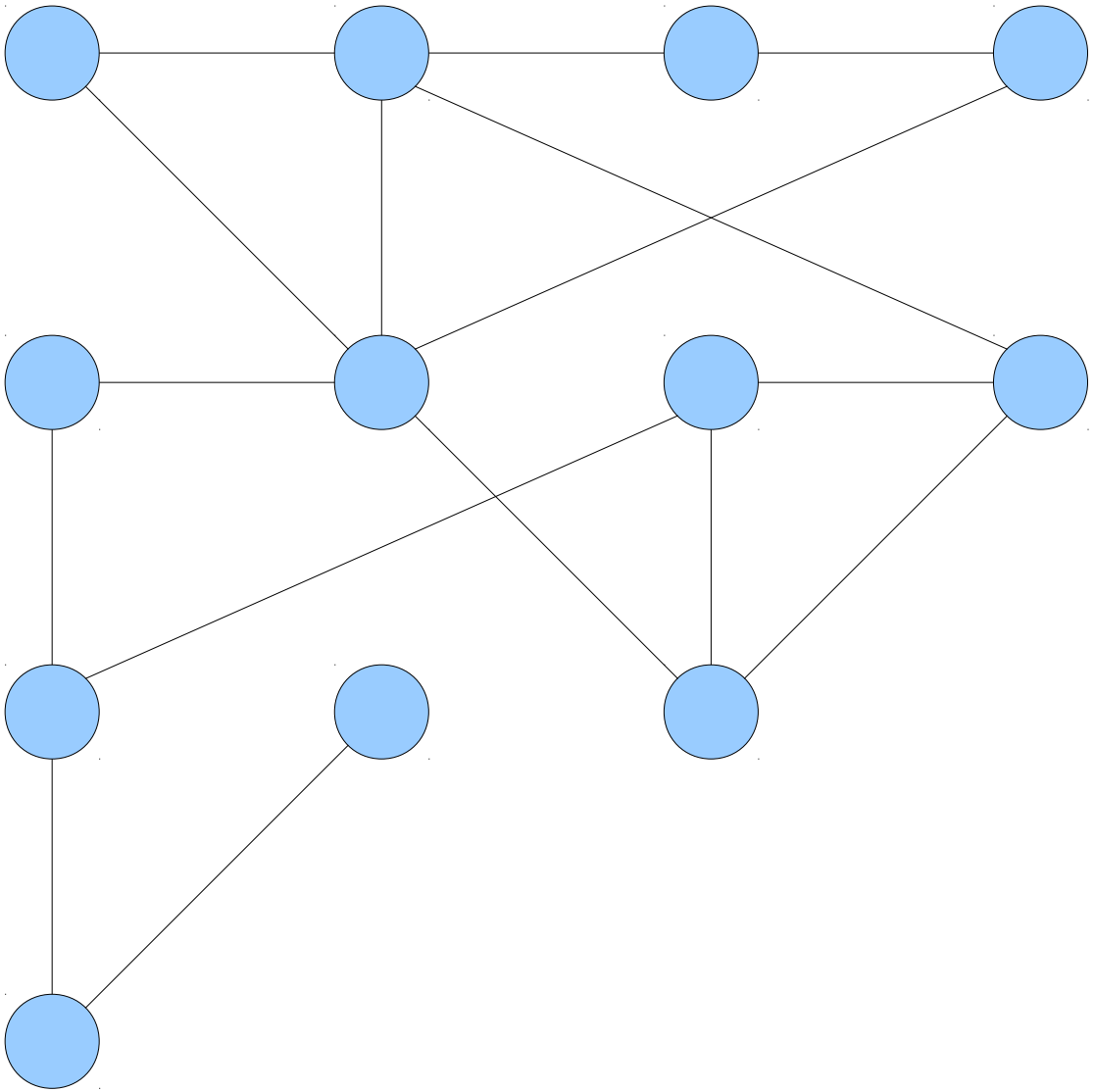


Depth-First Search

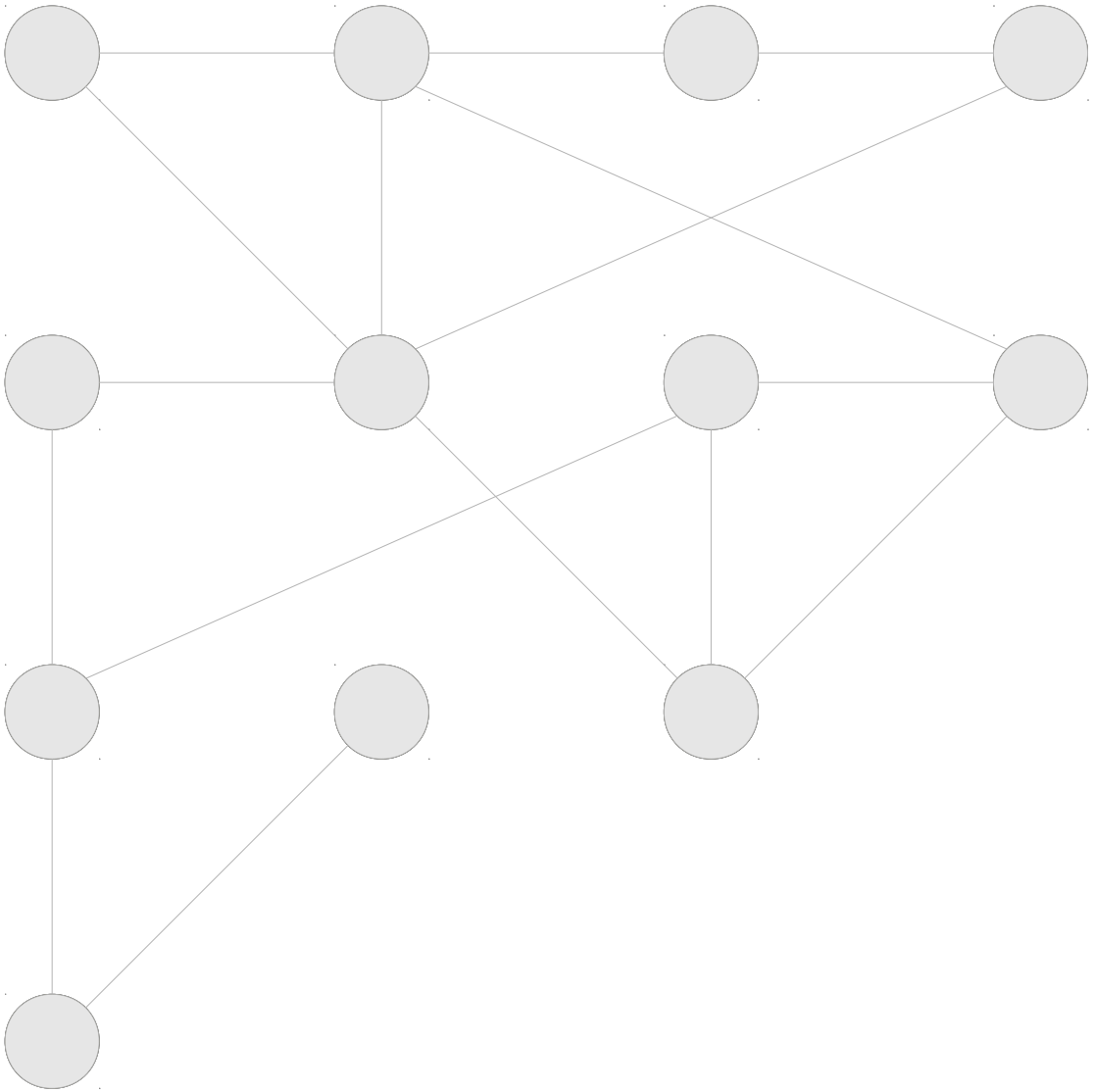
- To do a depth-first search (DFS) from a node *u*, do the following:
  - If *u* is already marked, stop.
  - Mark *u*.
  - For each neighbor *v* of *u:*
    - Recursively run DFS from *v*.
- The backtracking here is similar to the backtracking done in standard recursion.

Coding Depth-First Search

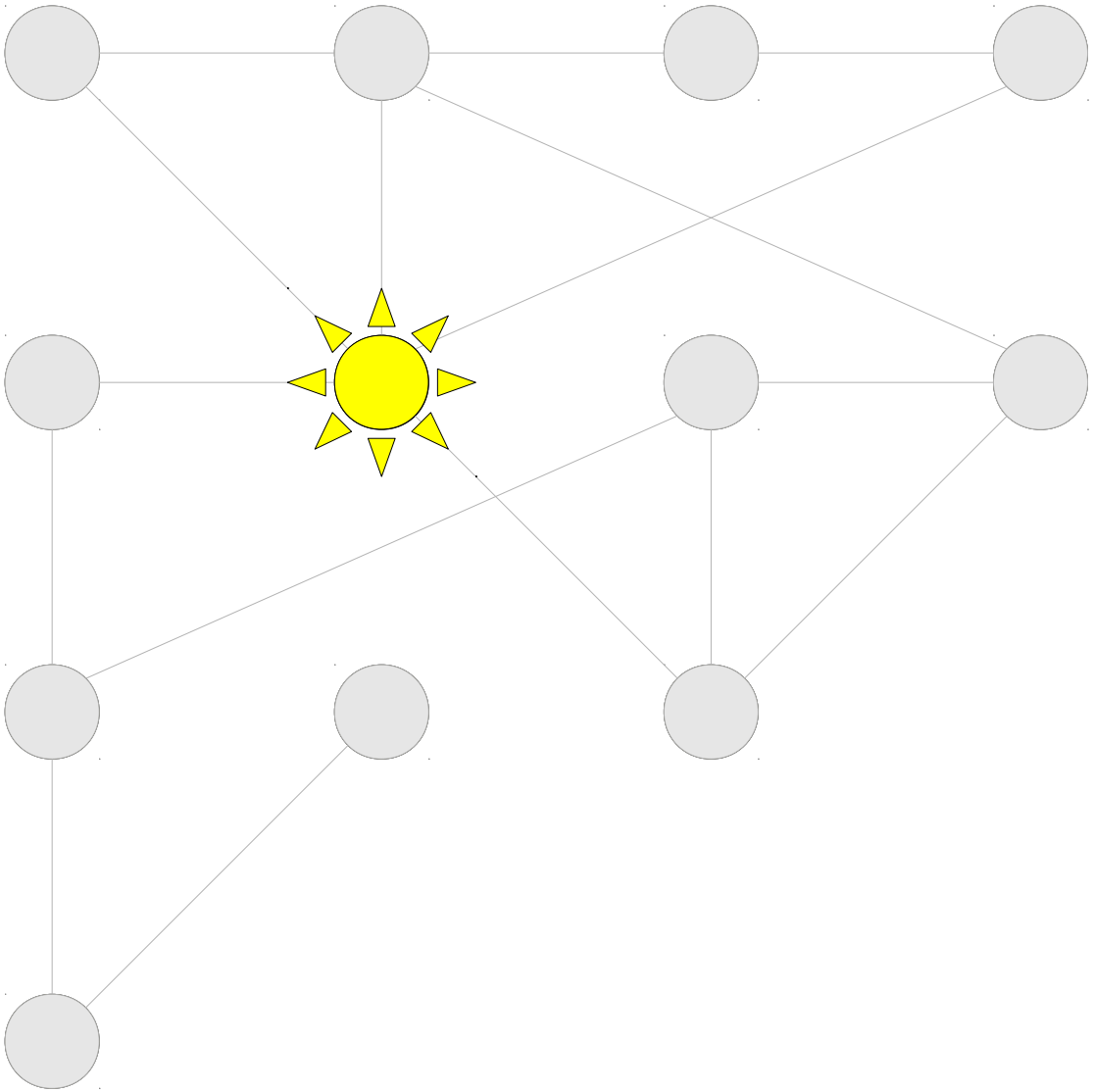
Depth-First Search



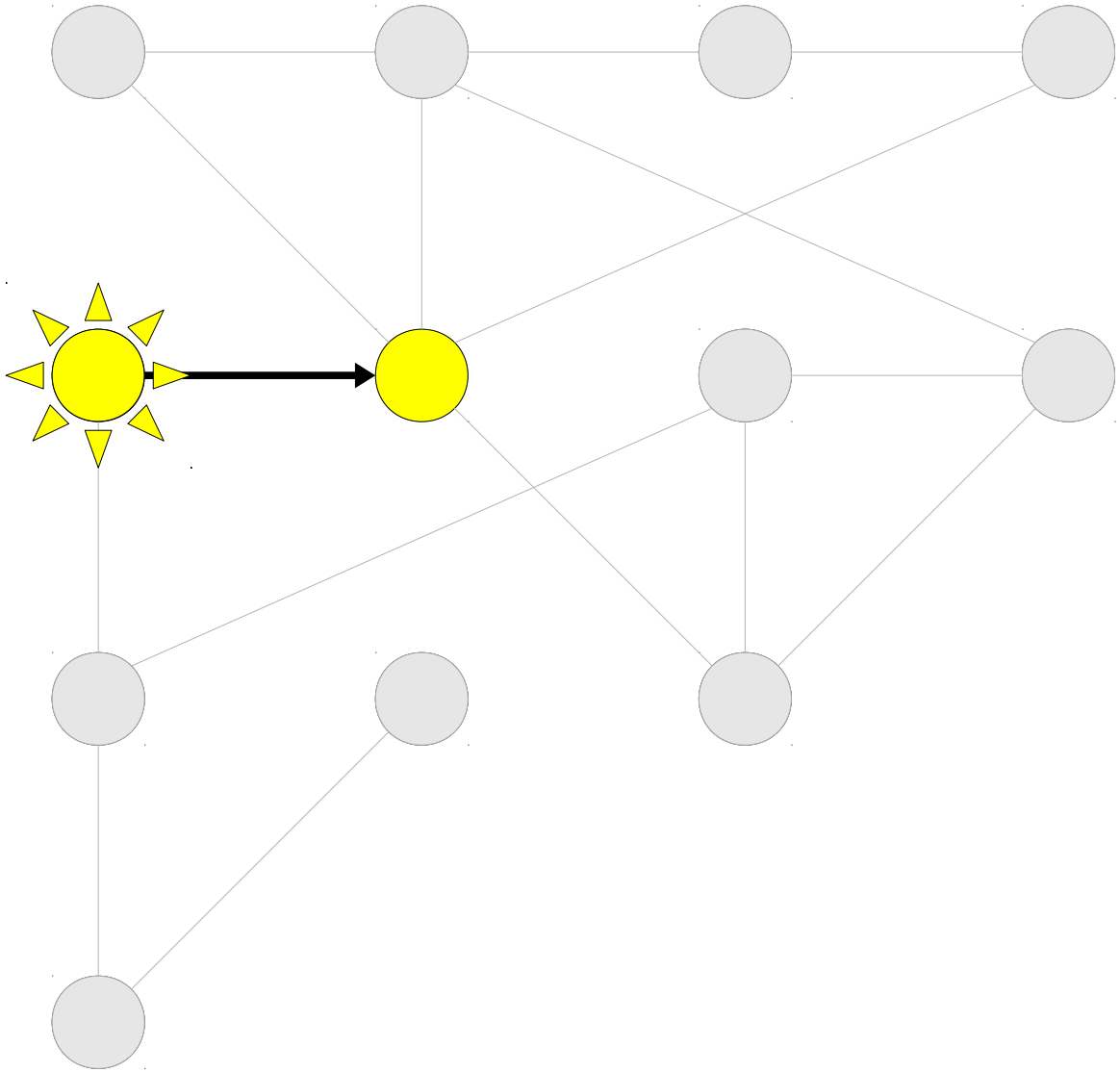
Depth-First Search



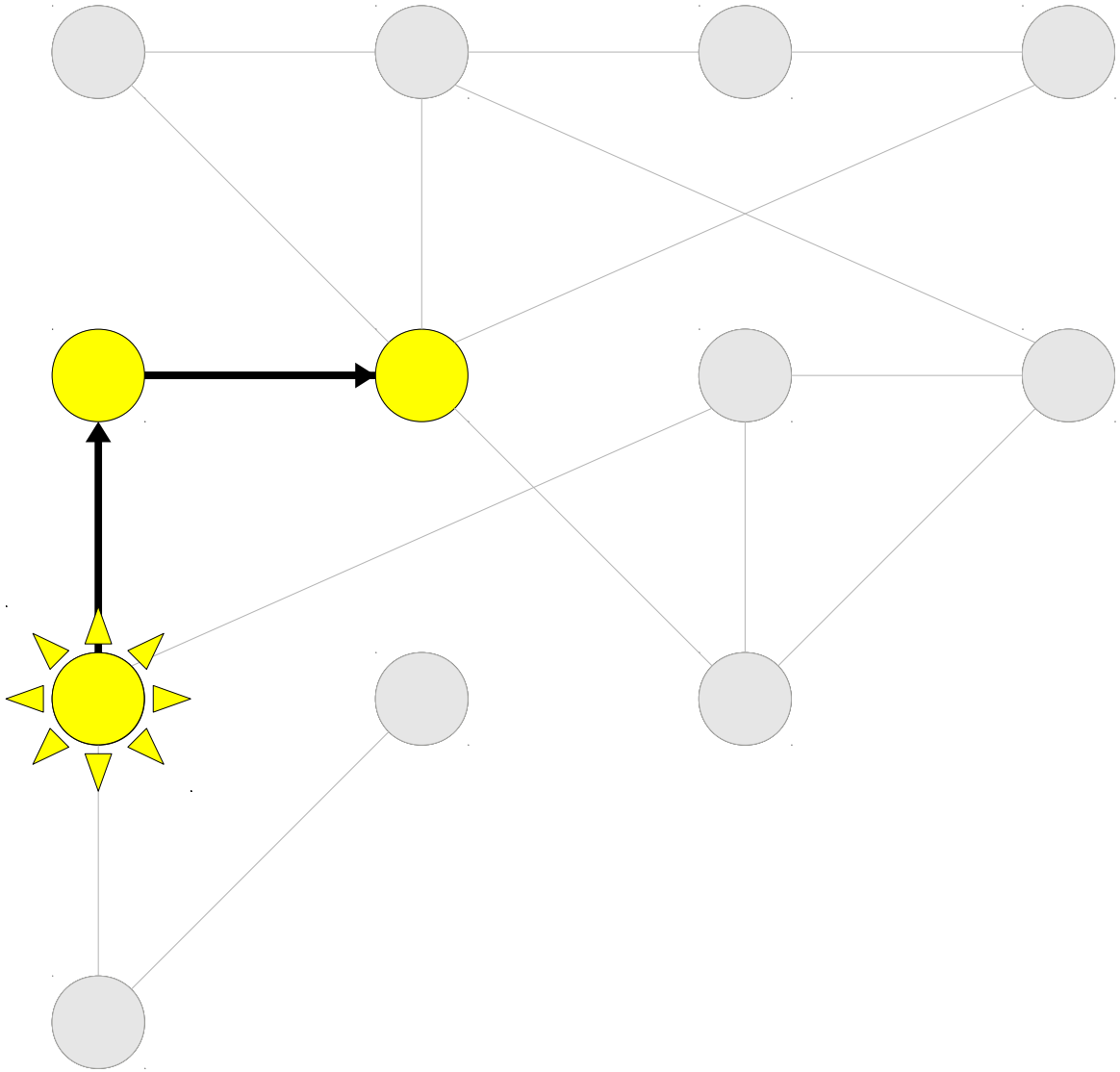
Depth-First Search



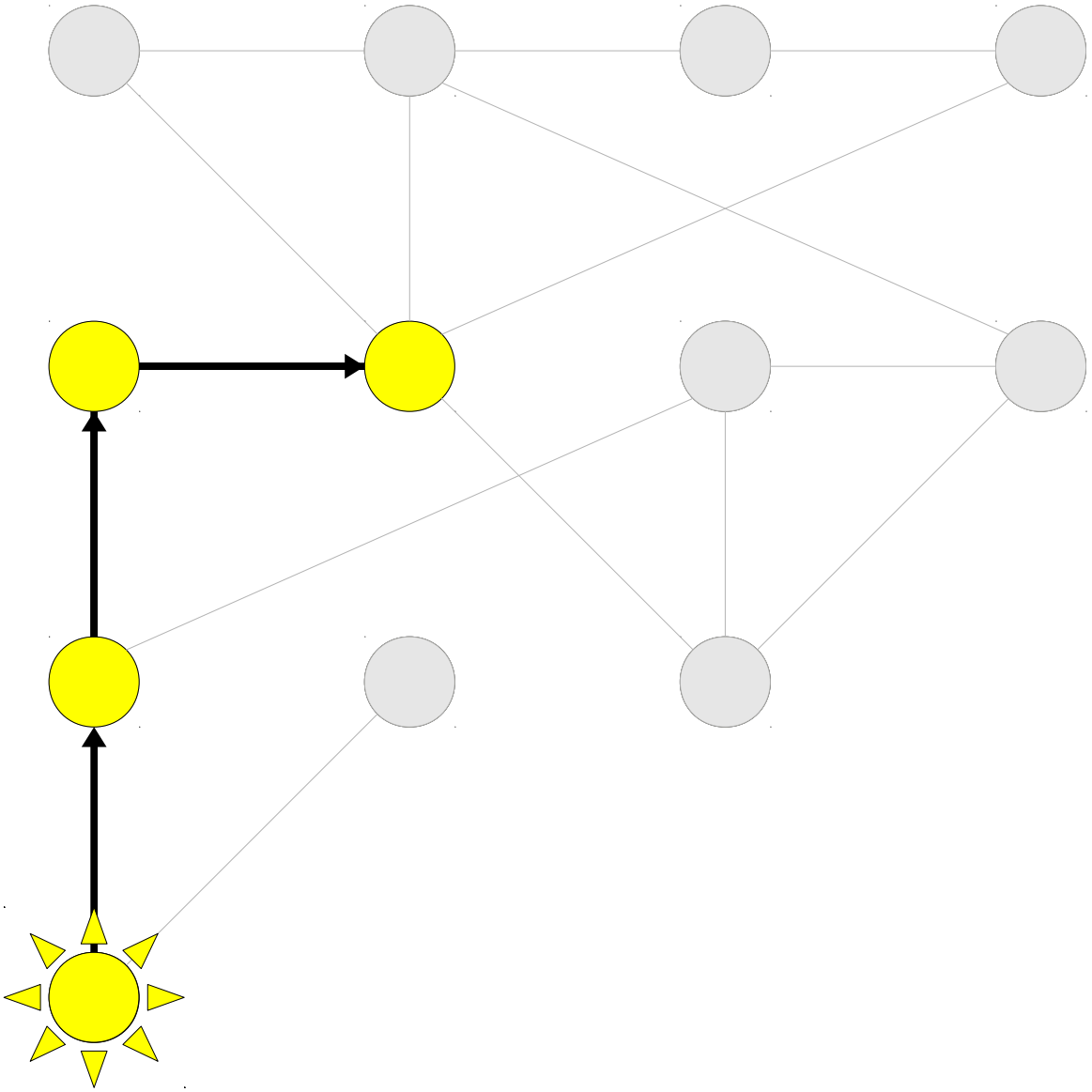
Depth-First Search



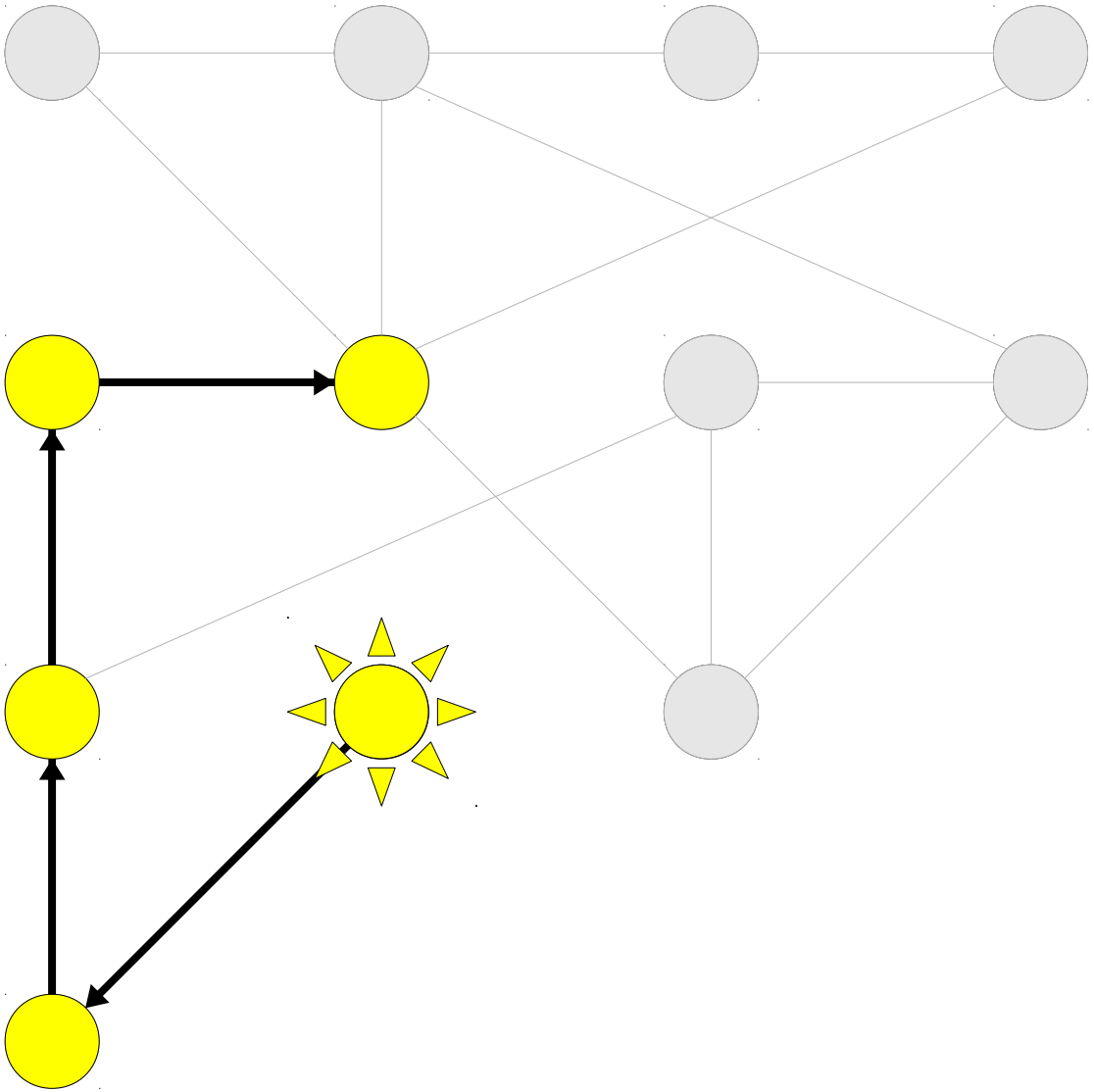
Depth-First Search



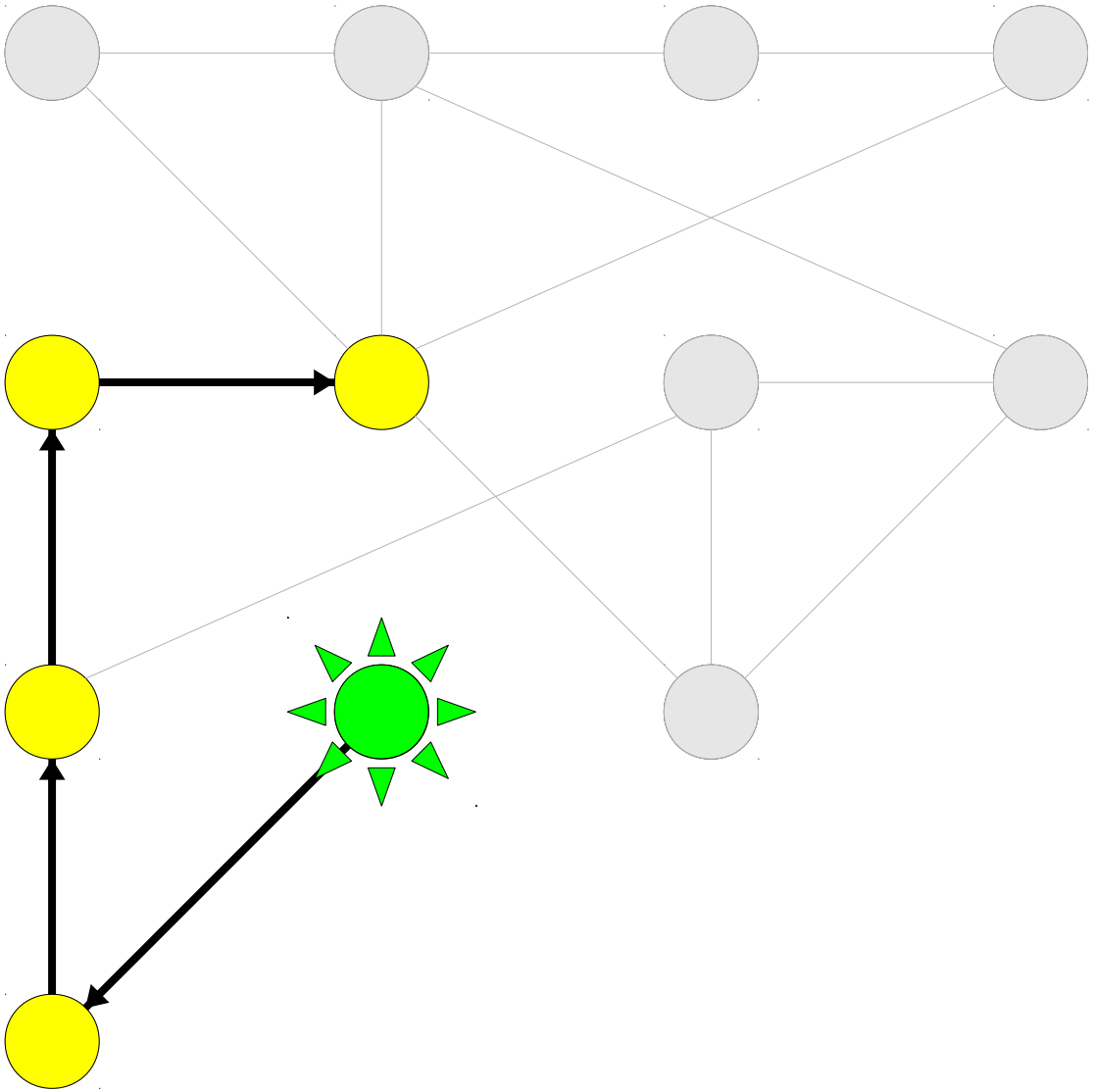
Depth-First Search



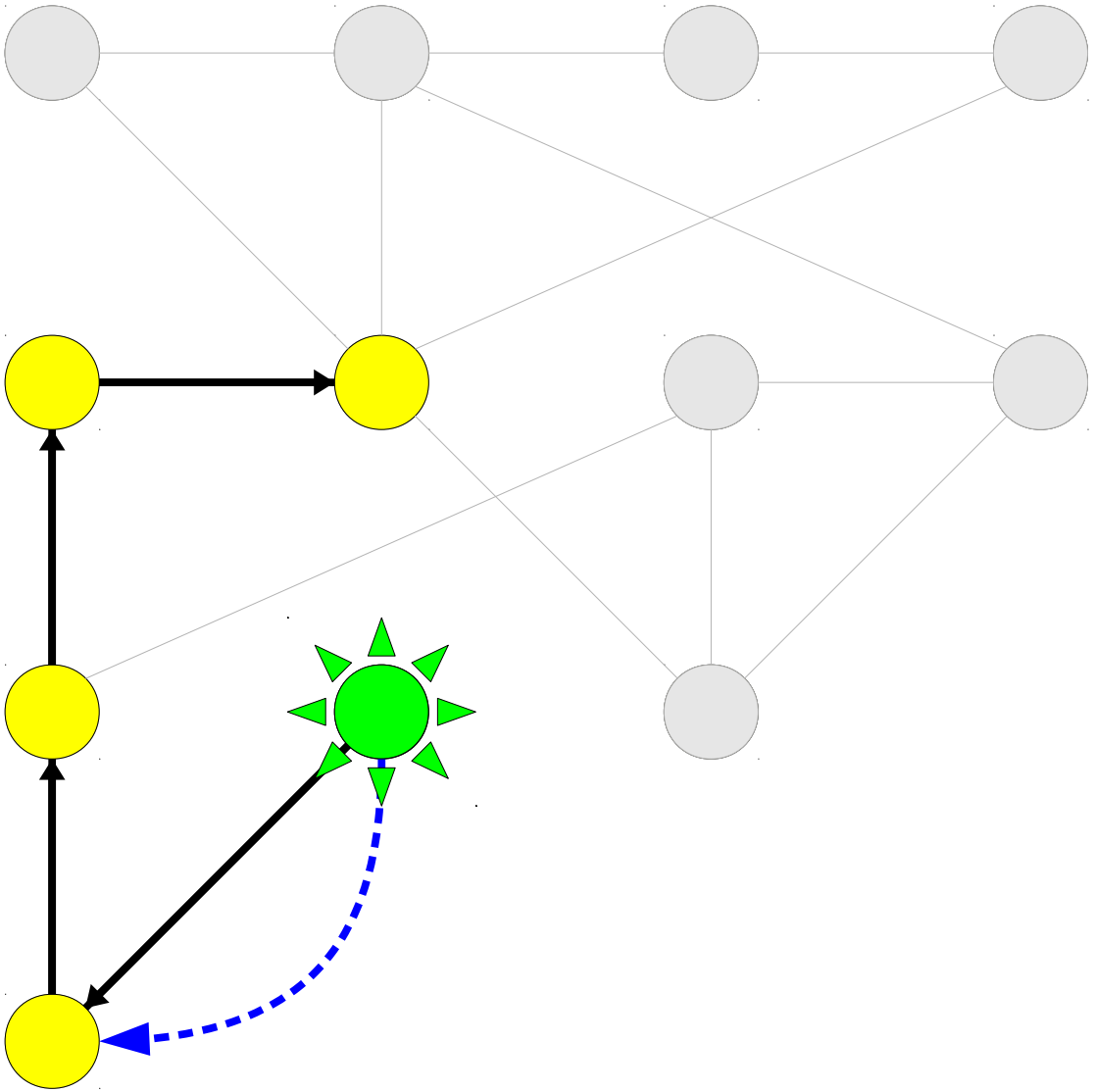
Depth-First Search



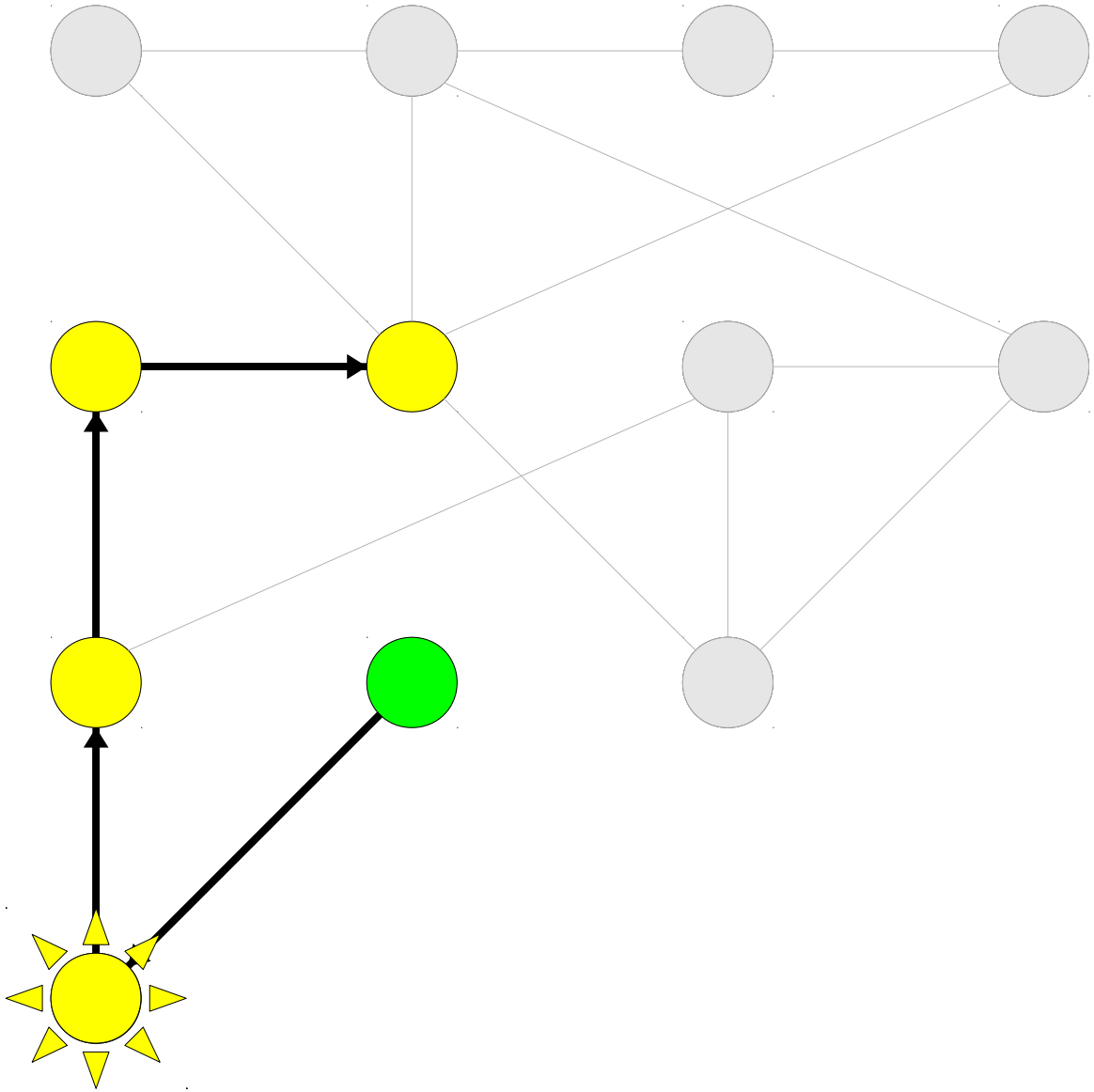
Depth-First Search



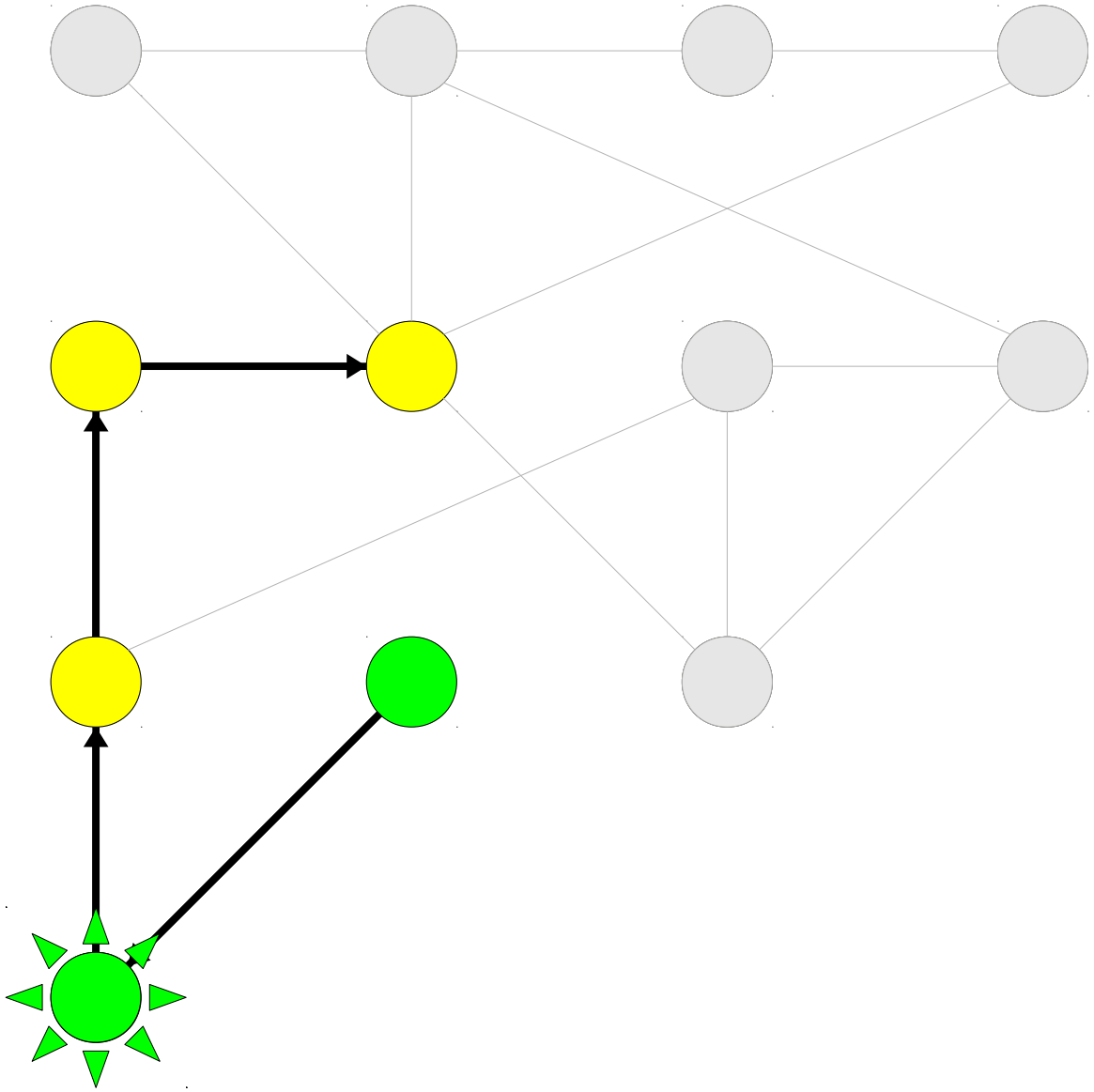
Depth-First Search



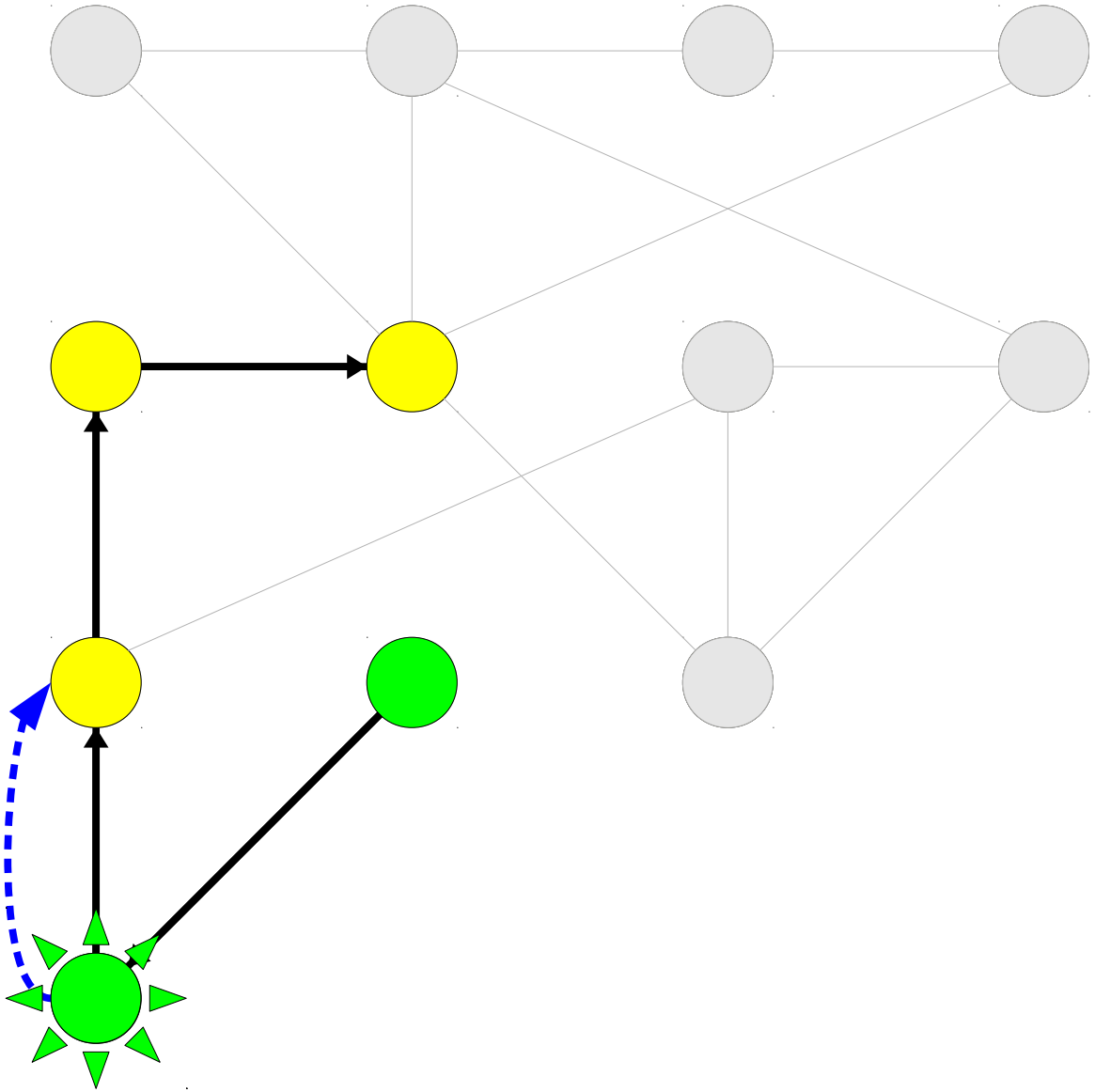
Depth-First Search



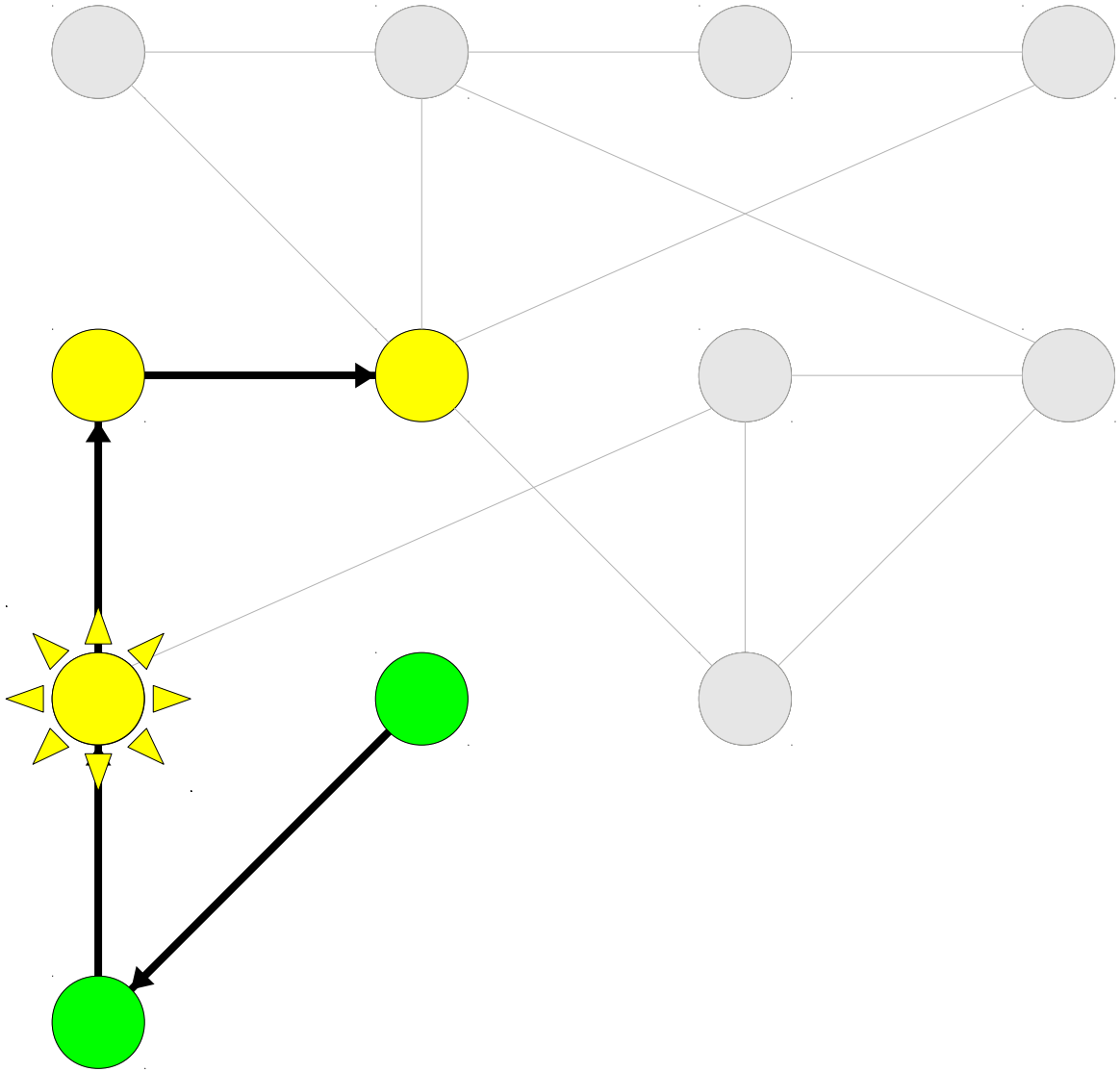
Depth-First Search



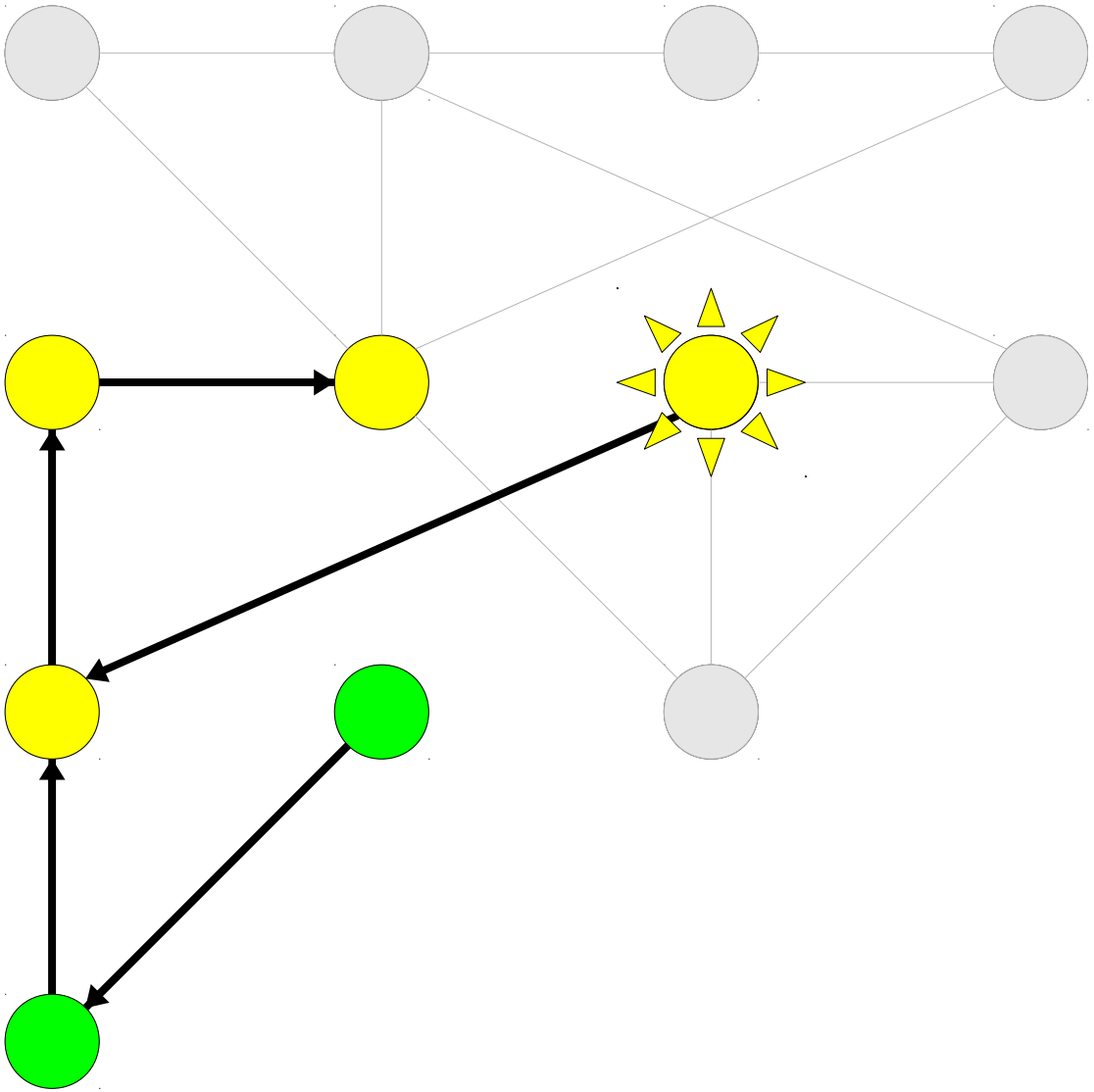
Depth-First Search



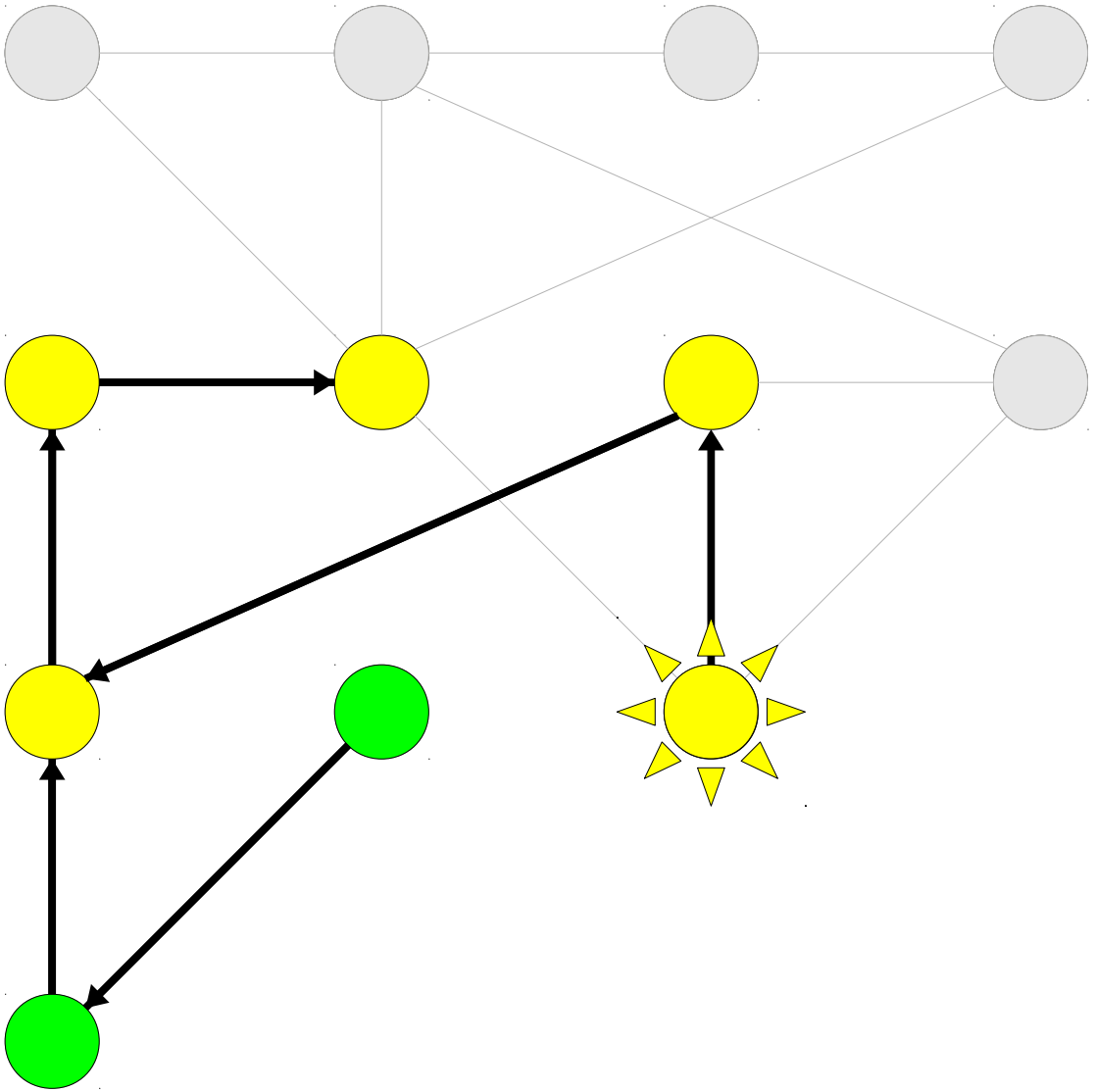
Depth-First Search



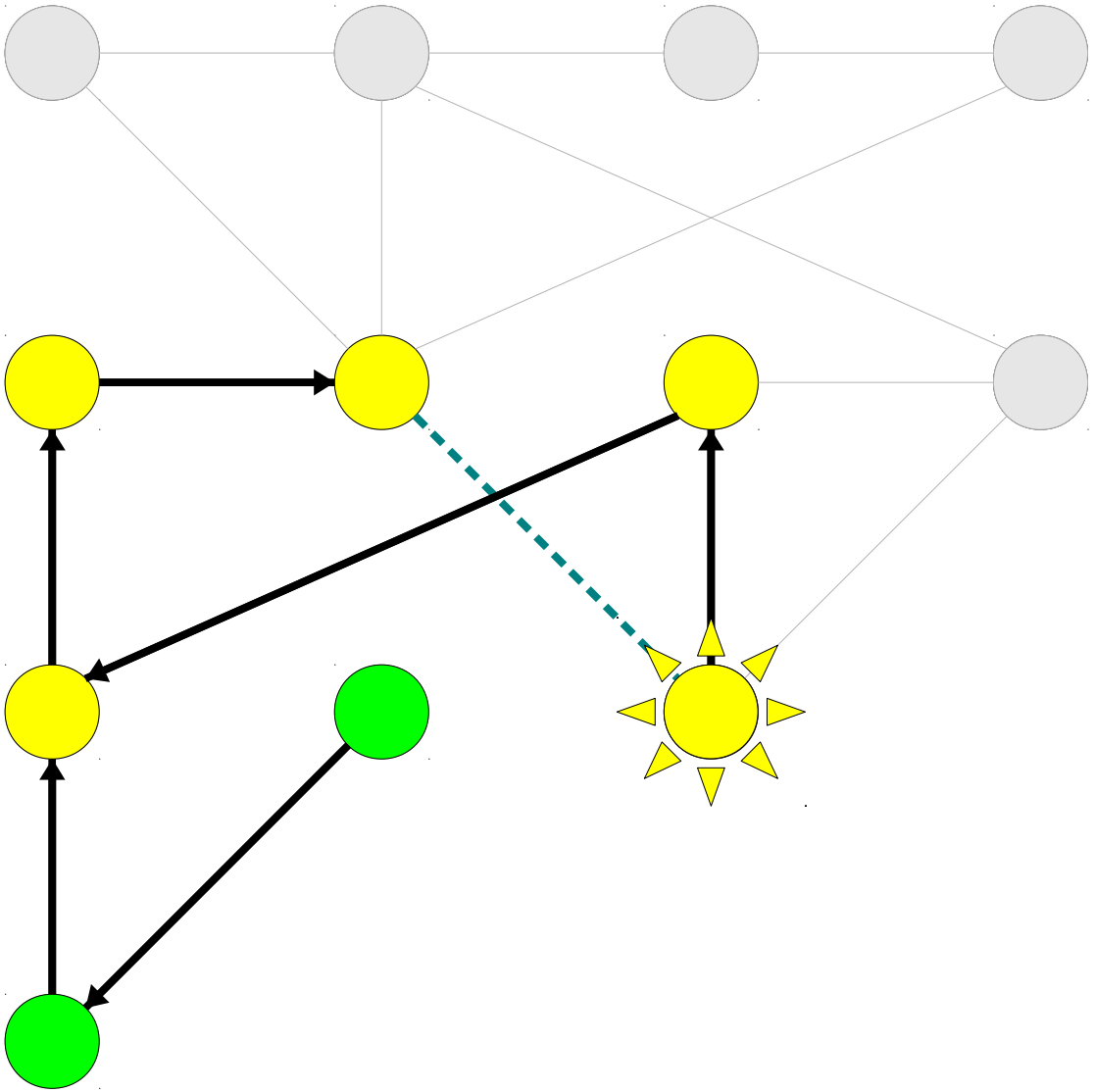
Depth-First Search



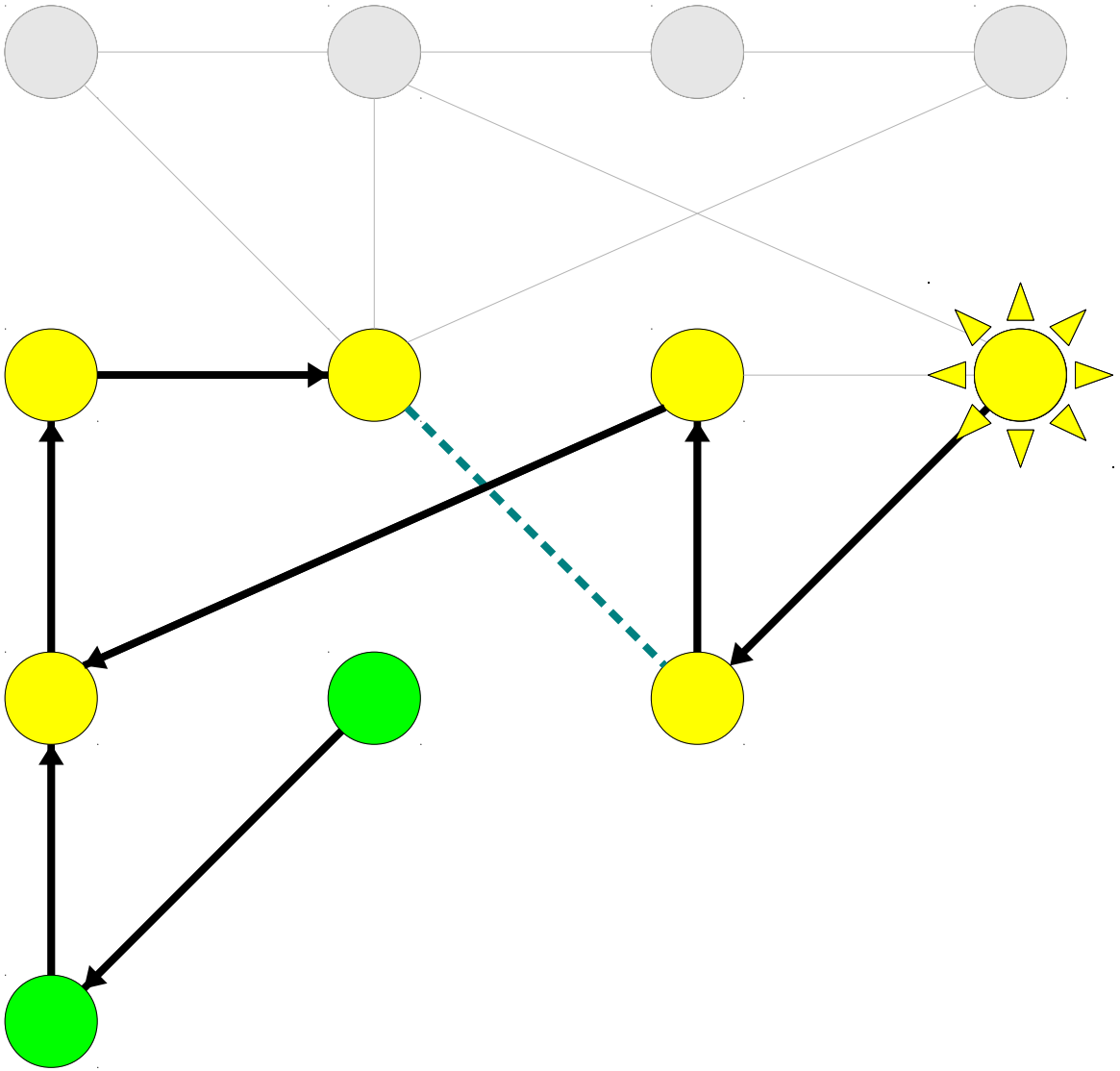
Depth-First Search



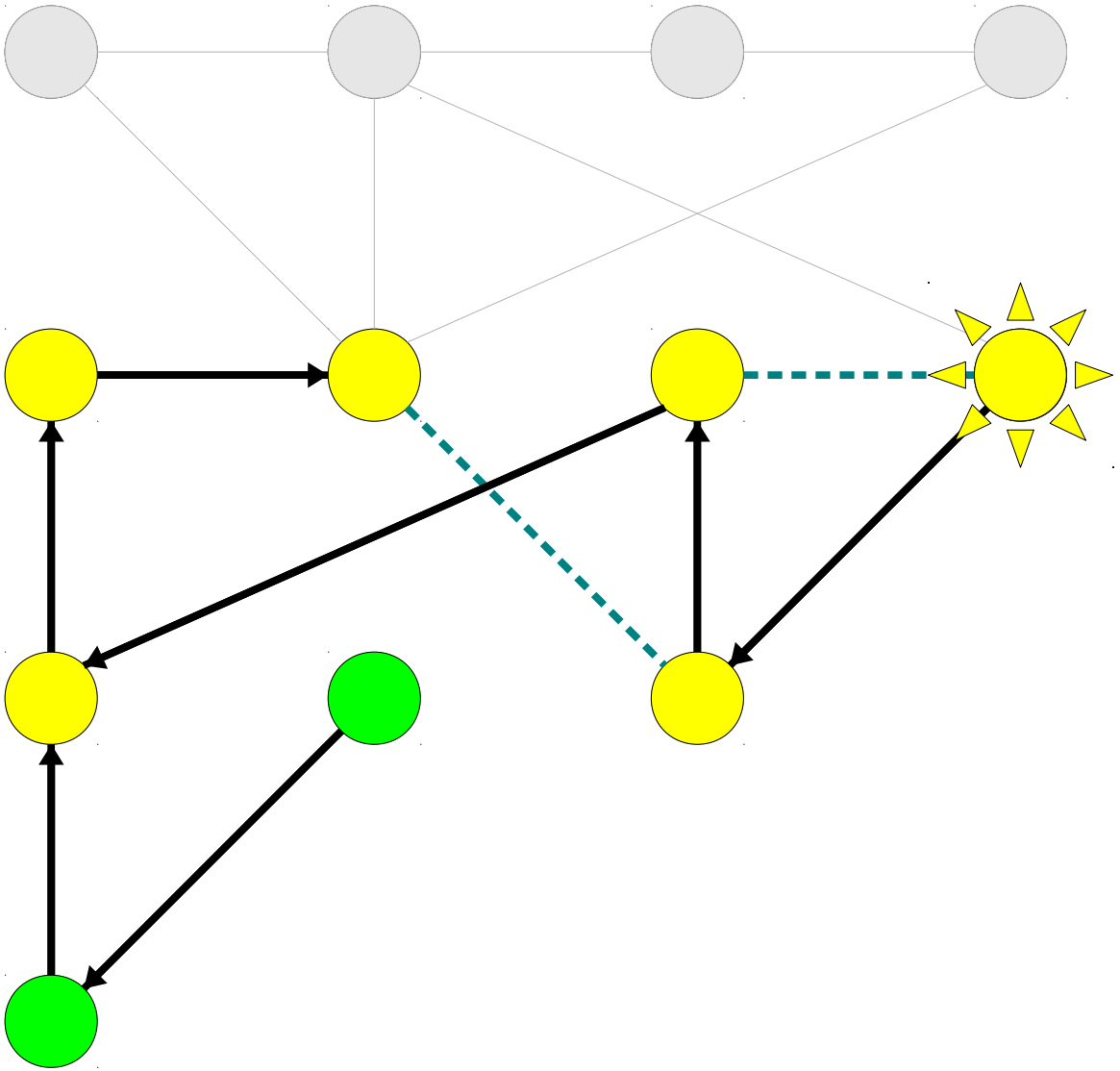
Depth-First Search



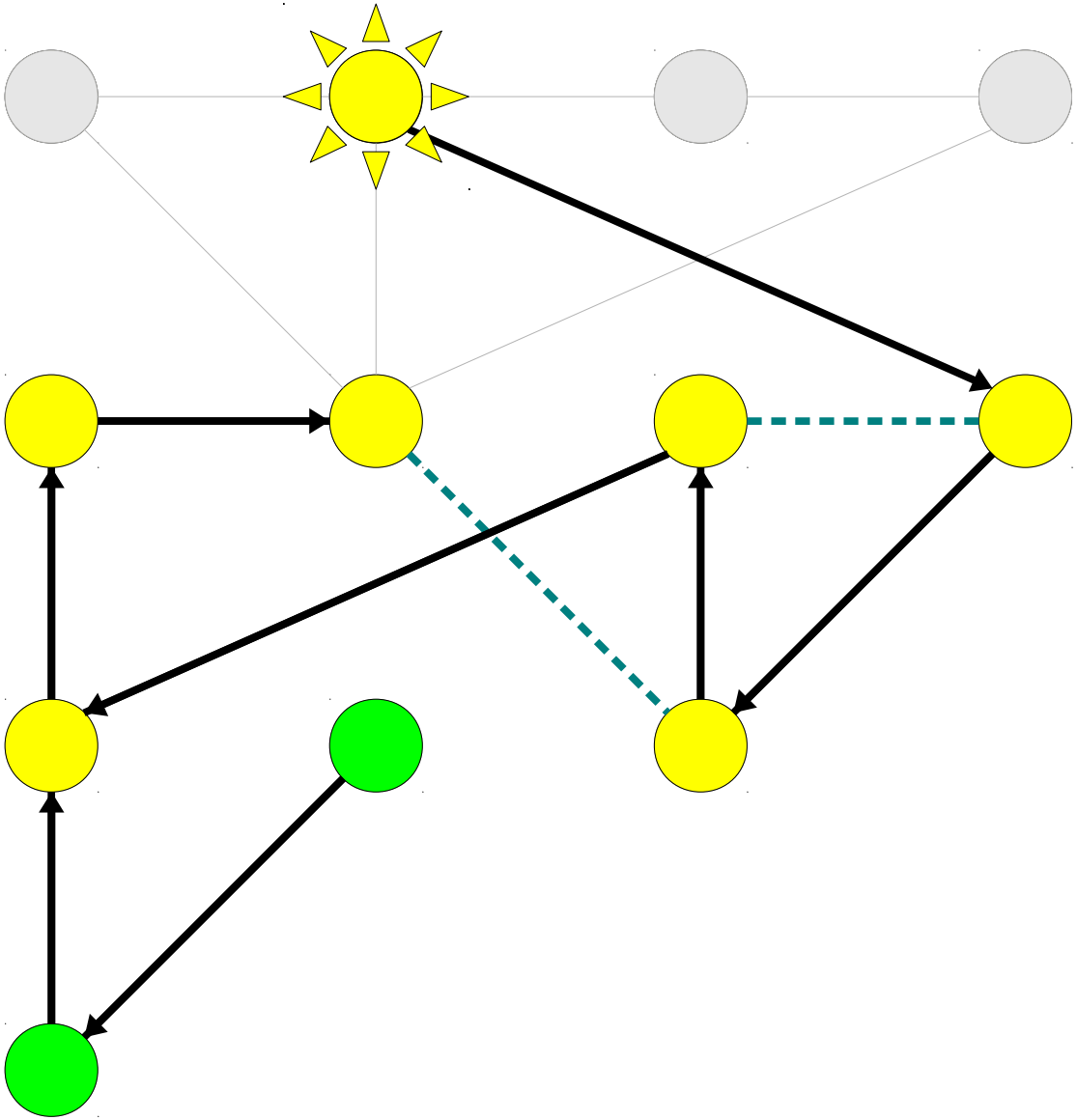
Depth-First Search



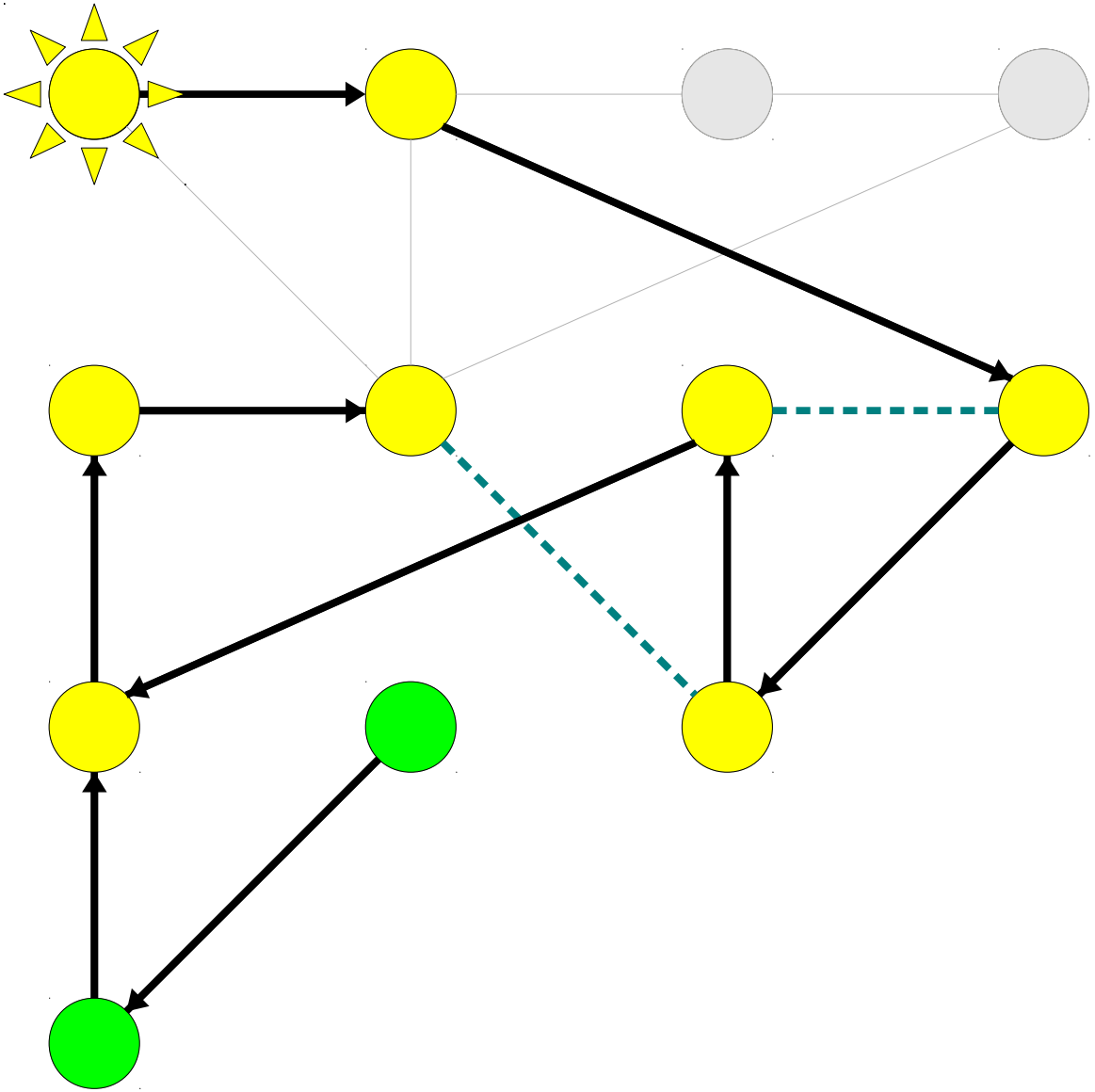
Depth-First Search



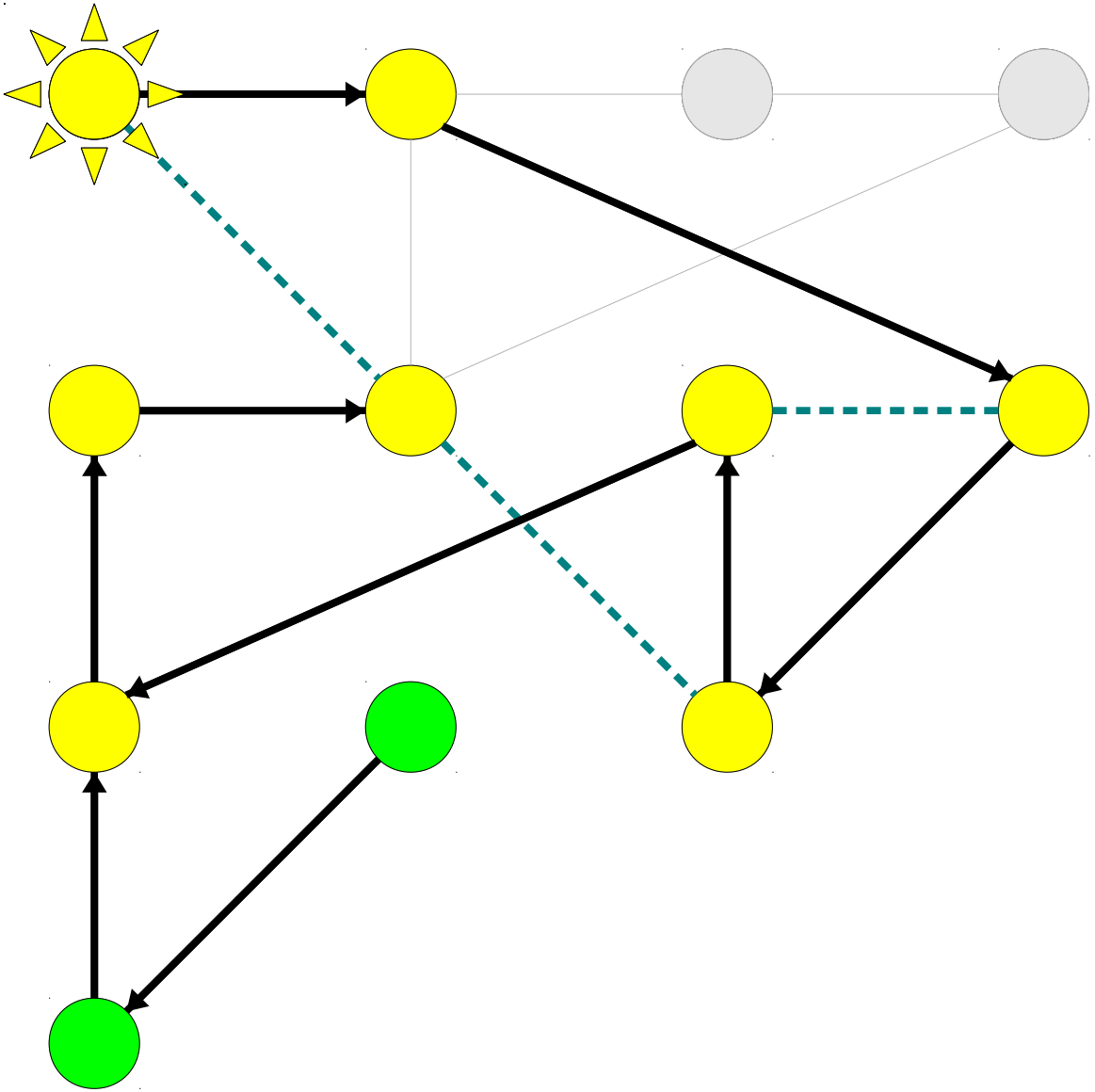
Depth-First Search



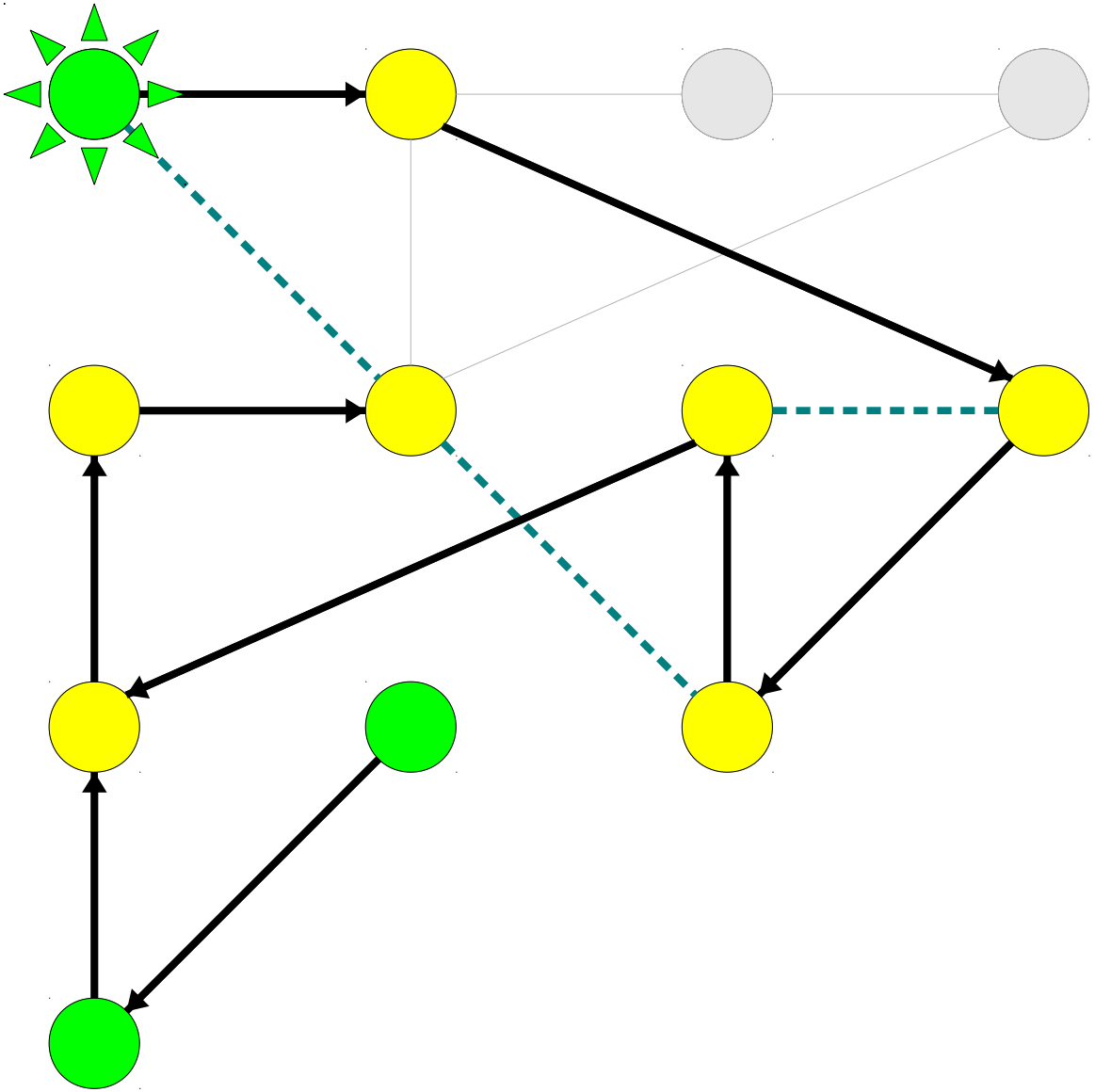
Depth-First Search



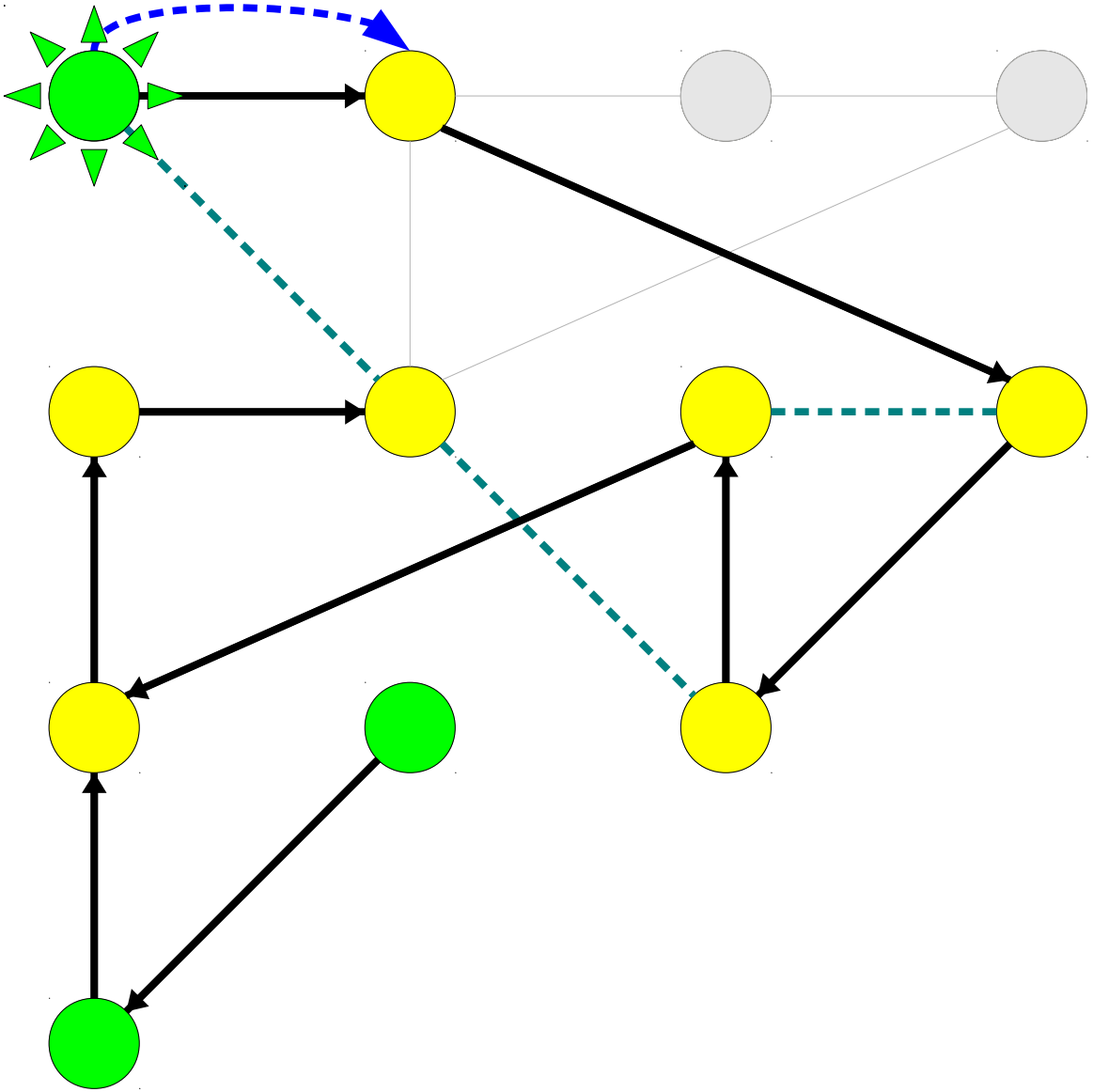
Depth-First Search



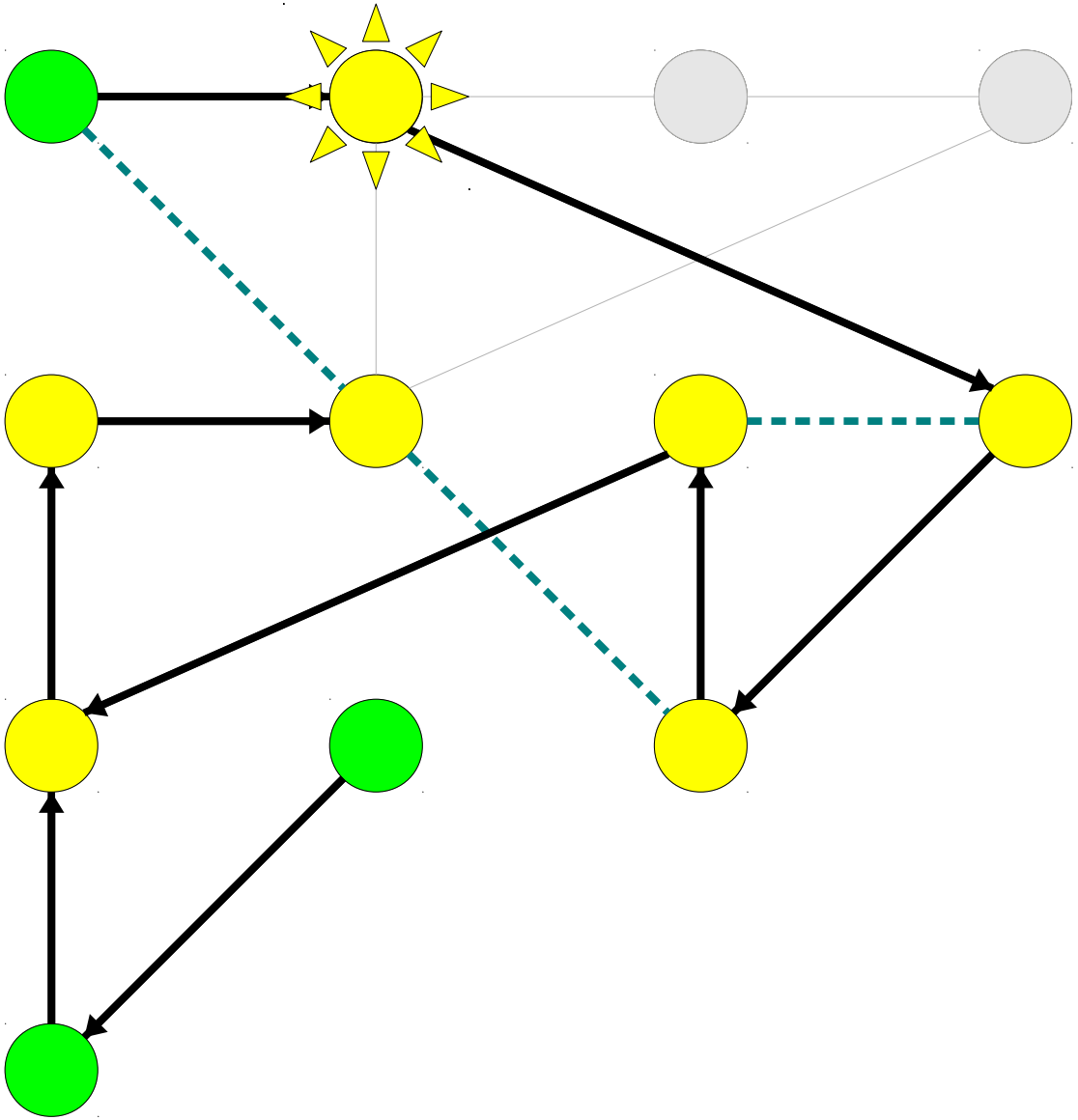
Depth-First Search



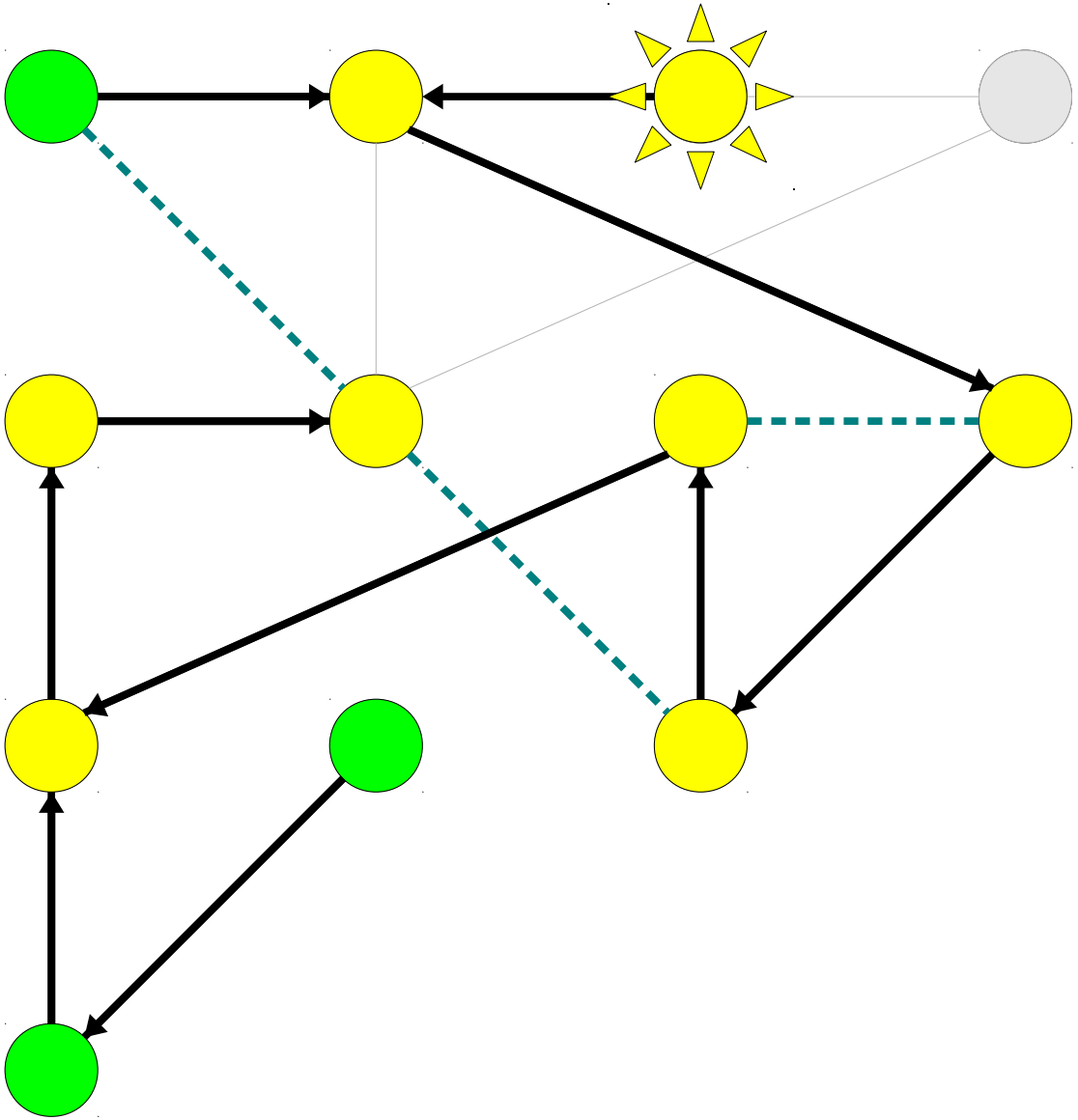
Depth-First Search



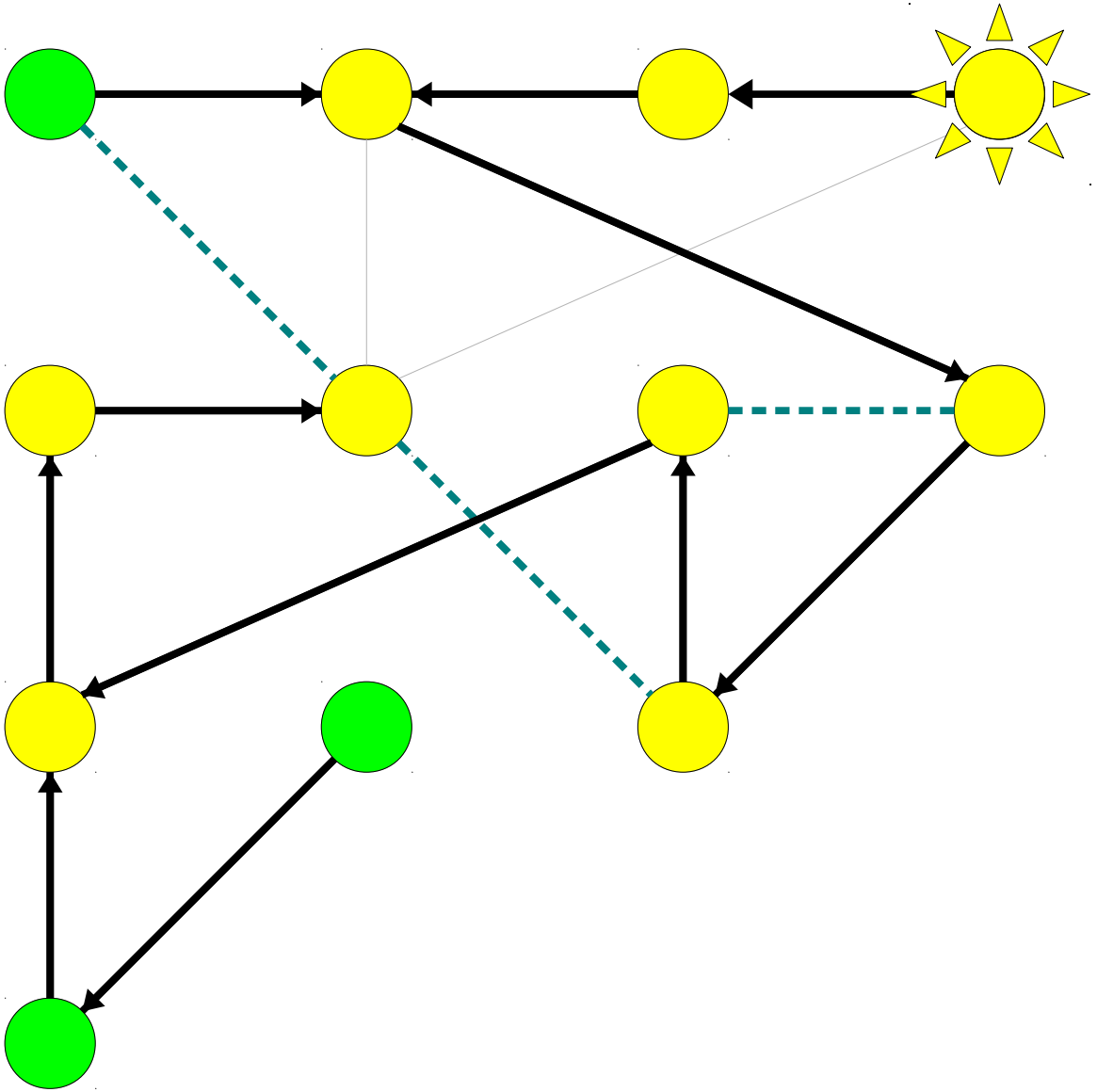
Depth-First Search



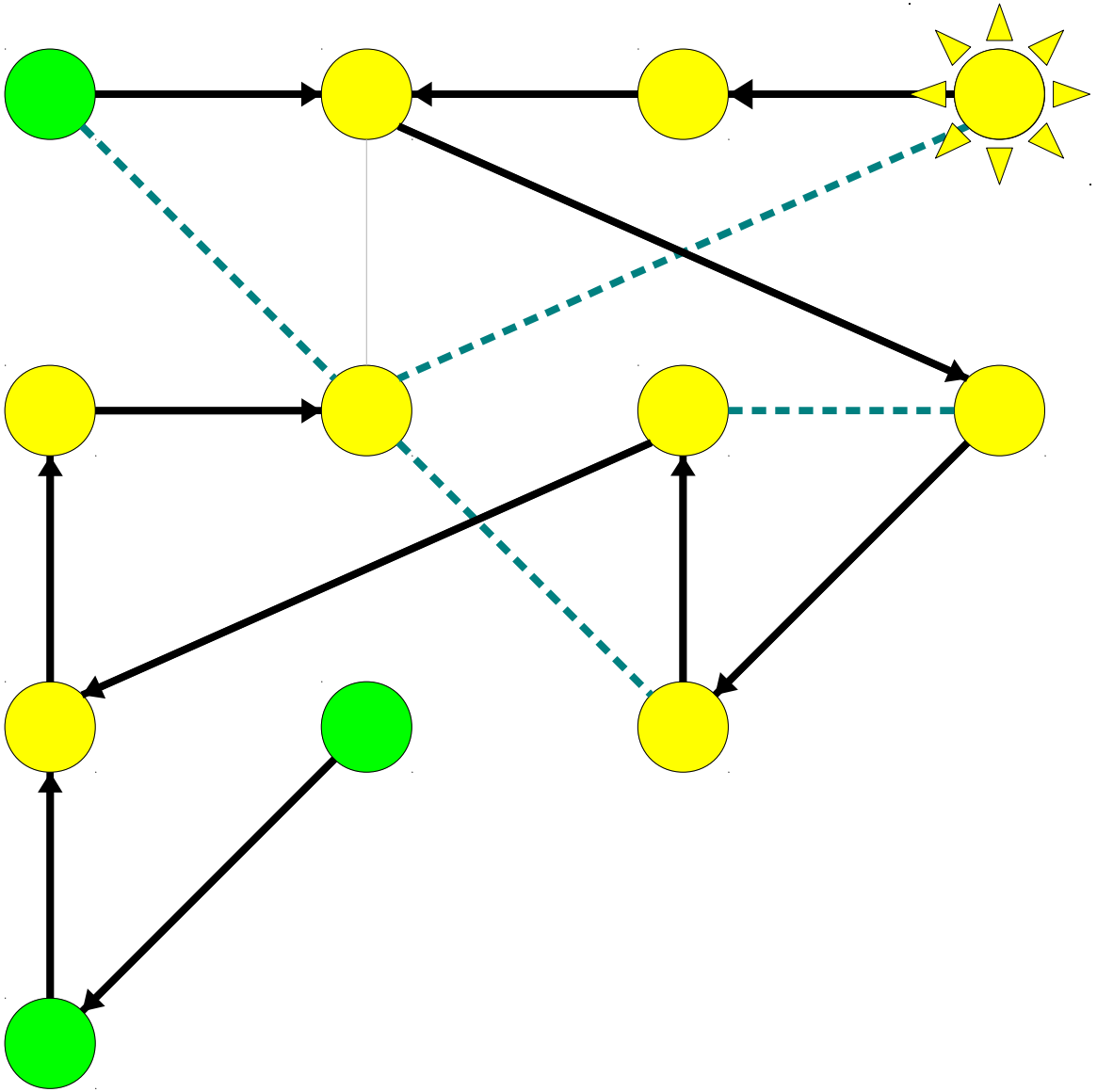
Depth-First Search



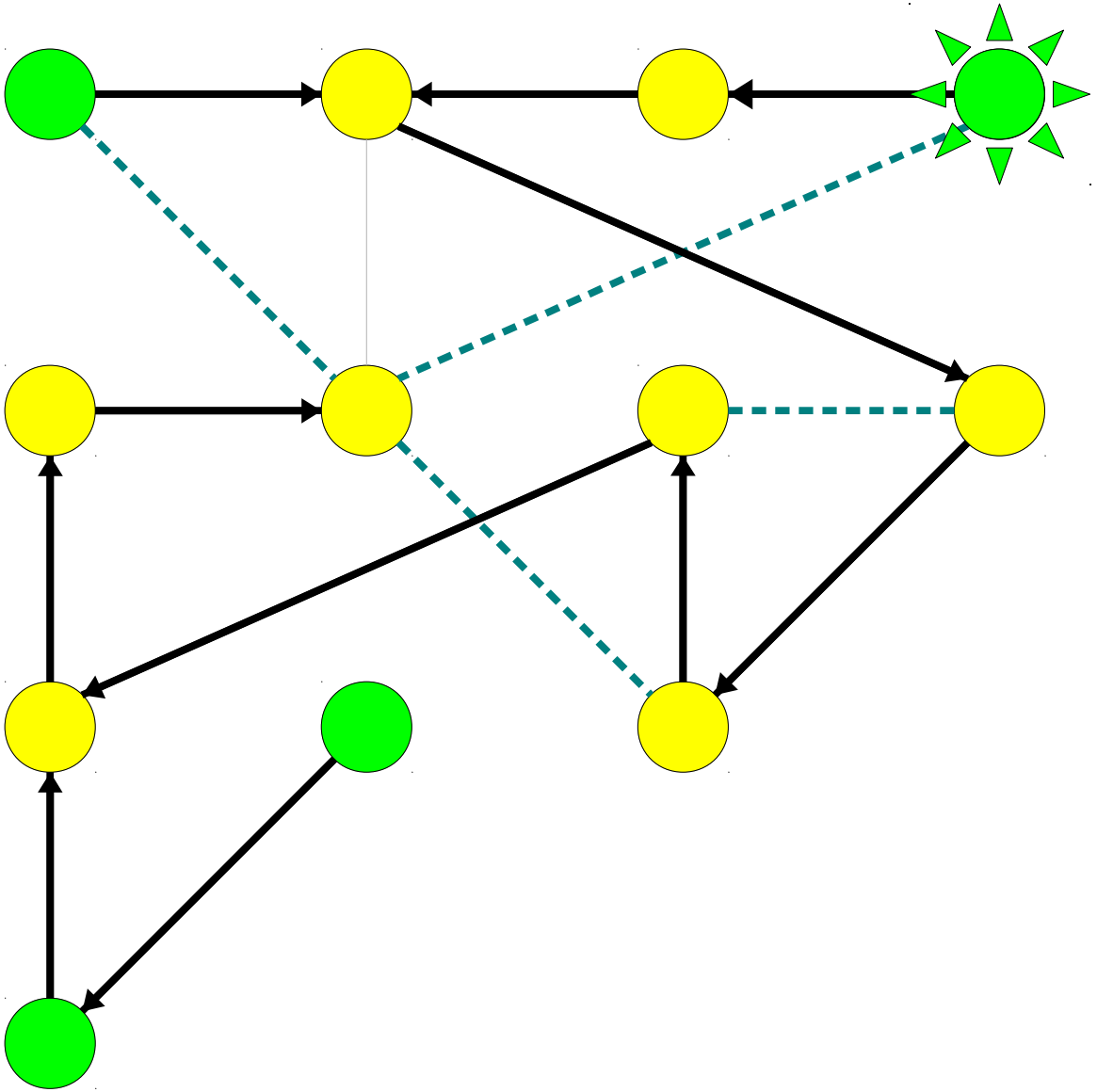
Depth-First Search



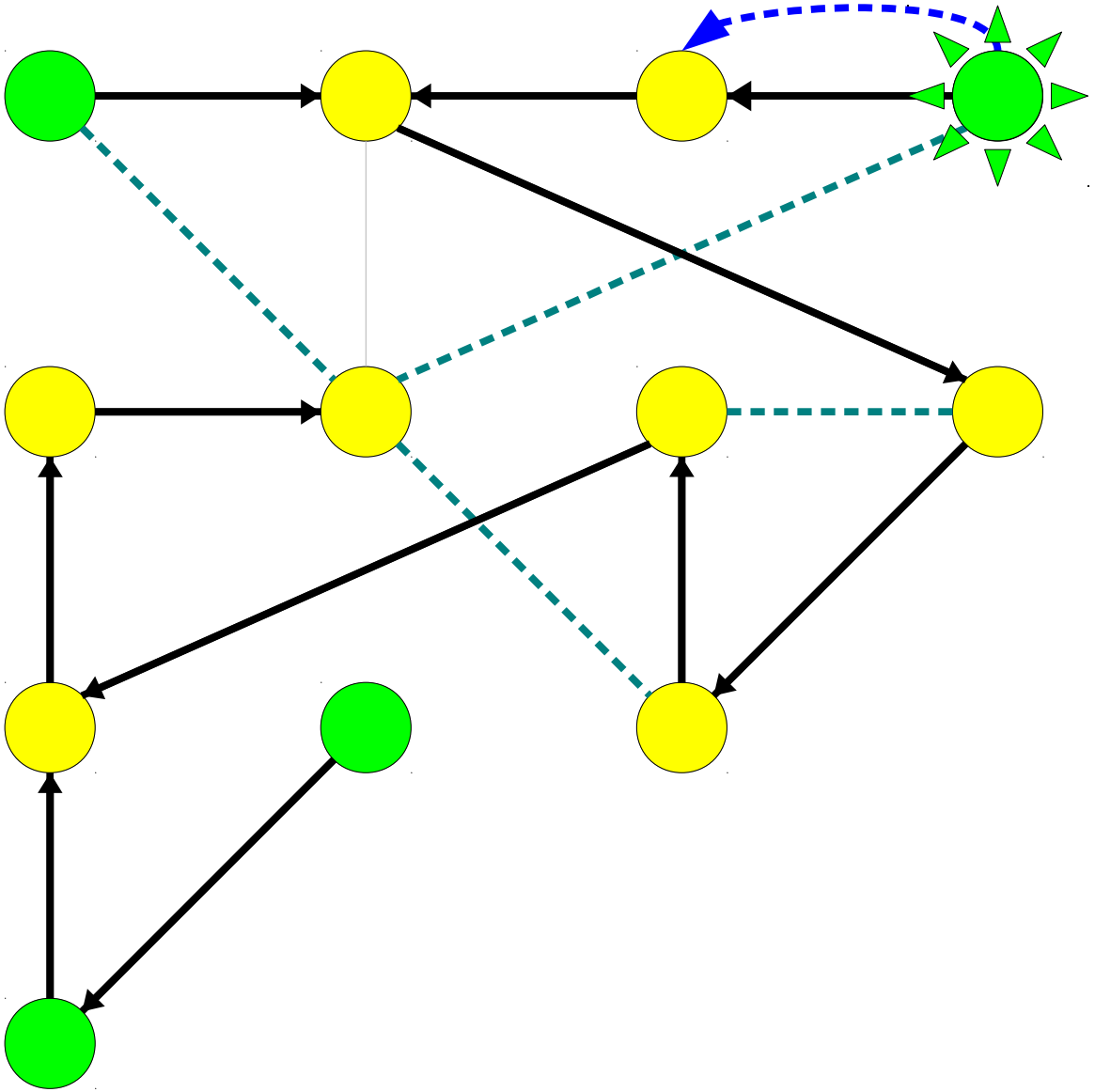
Depth-First Search



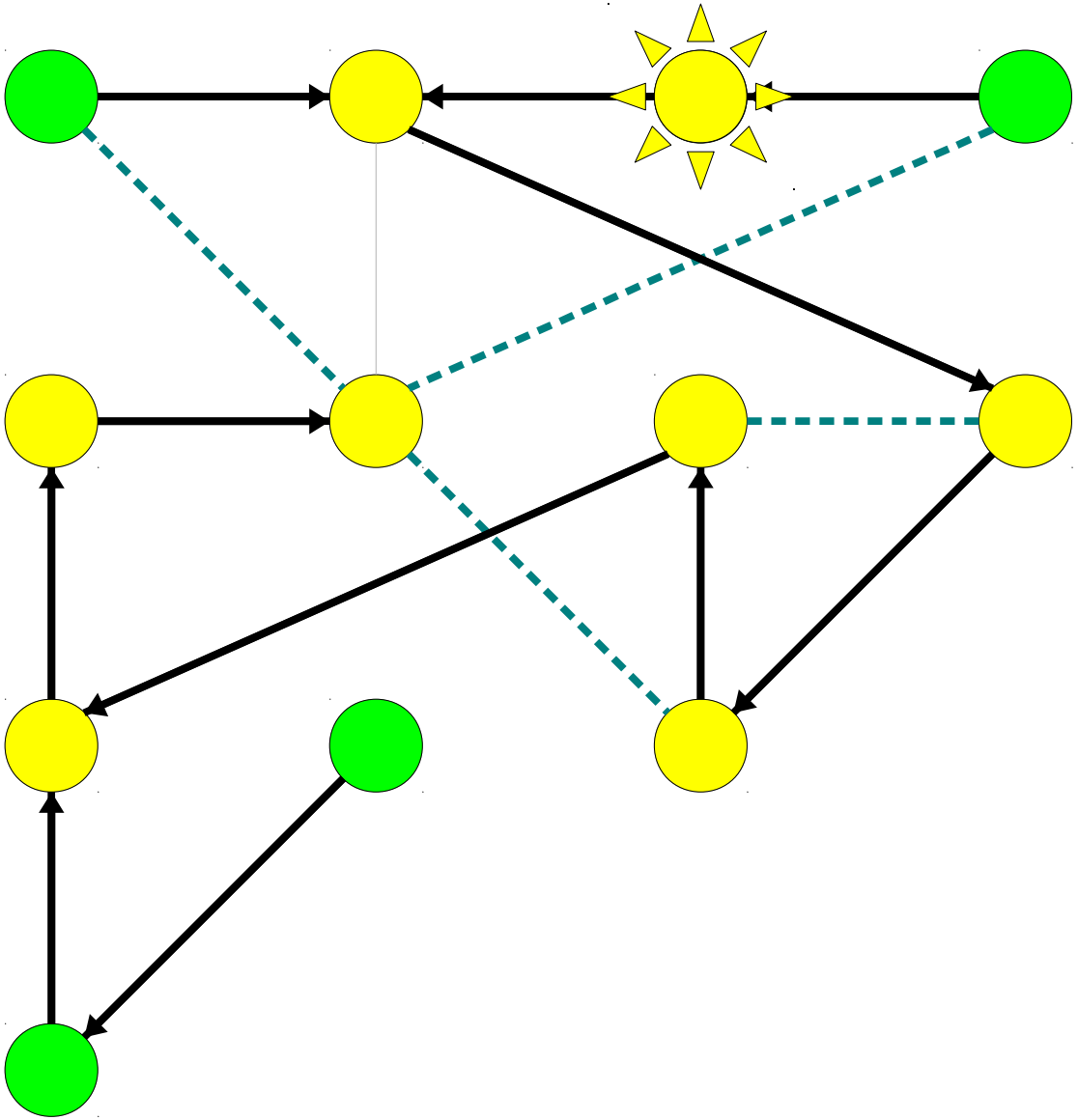
Depth-First Search



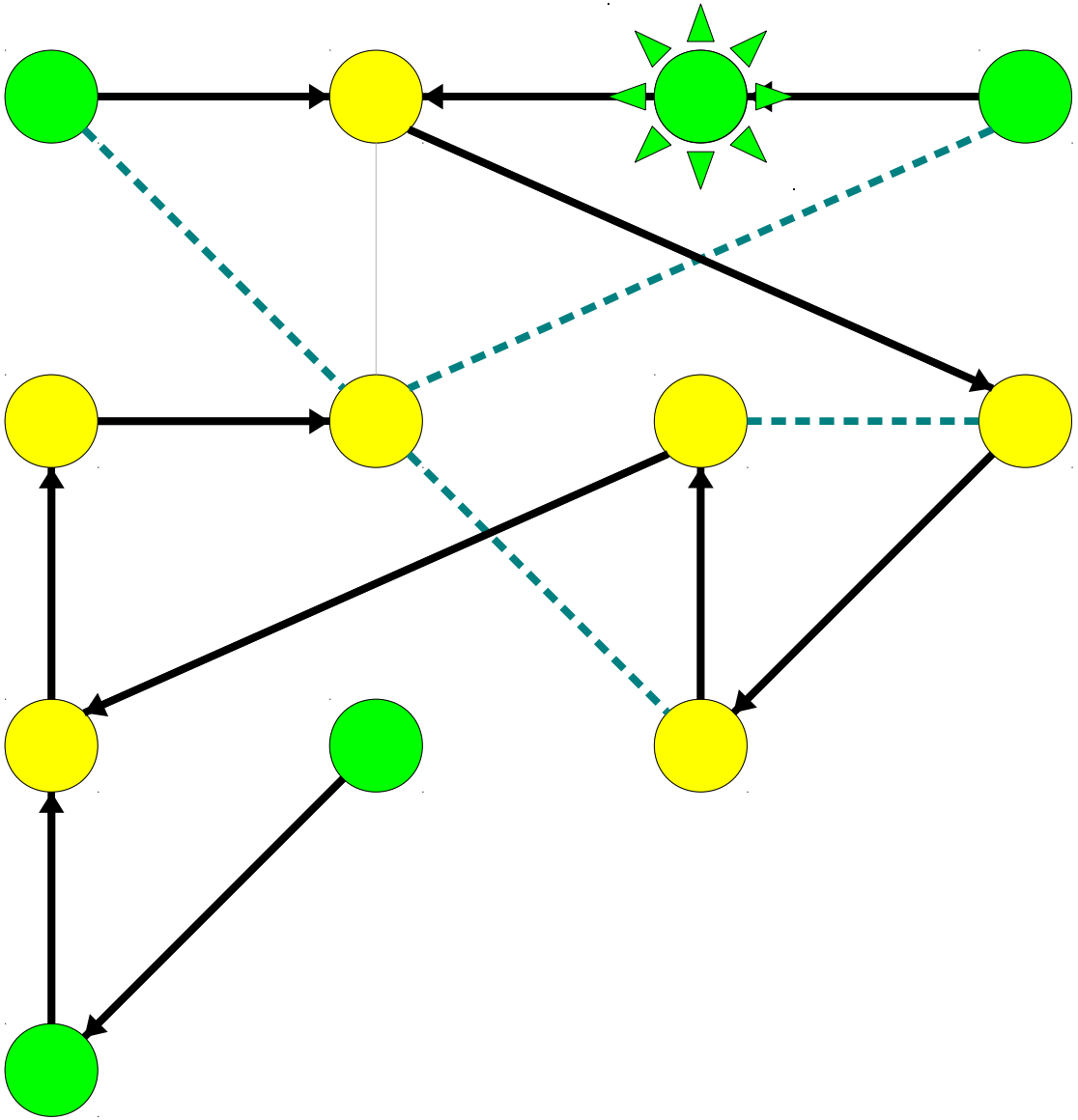
Depth-First Search



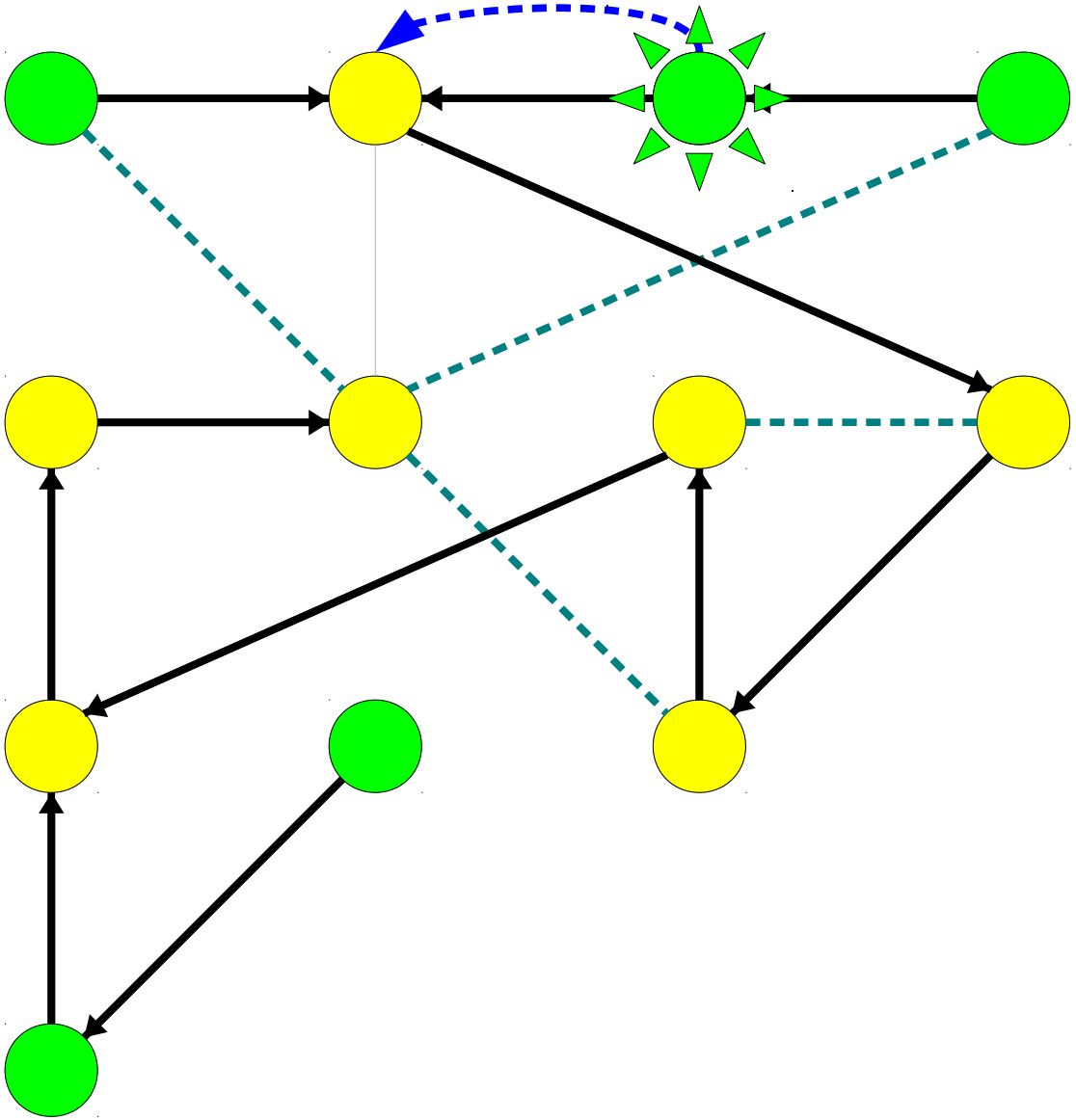
Depth-First Search



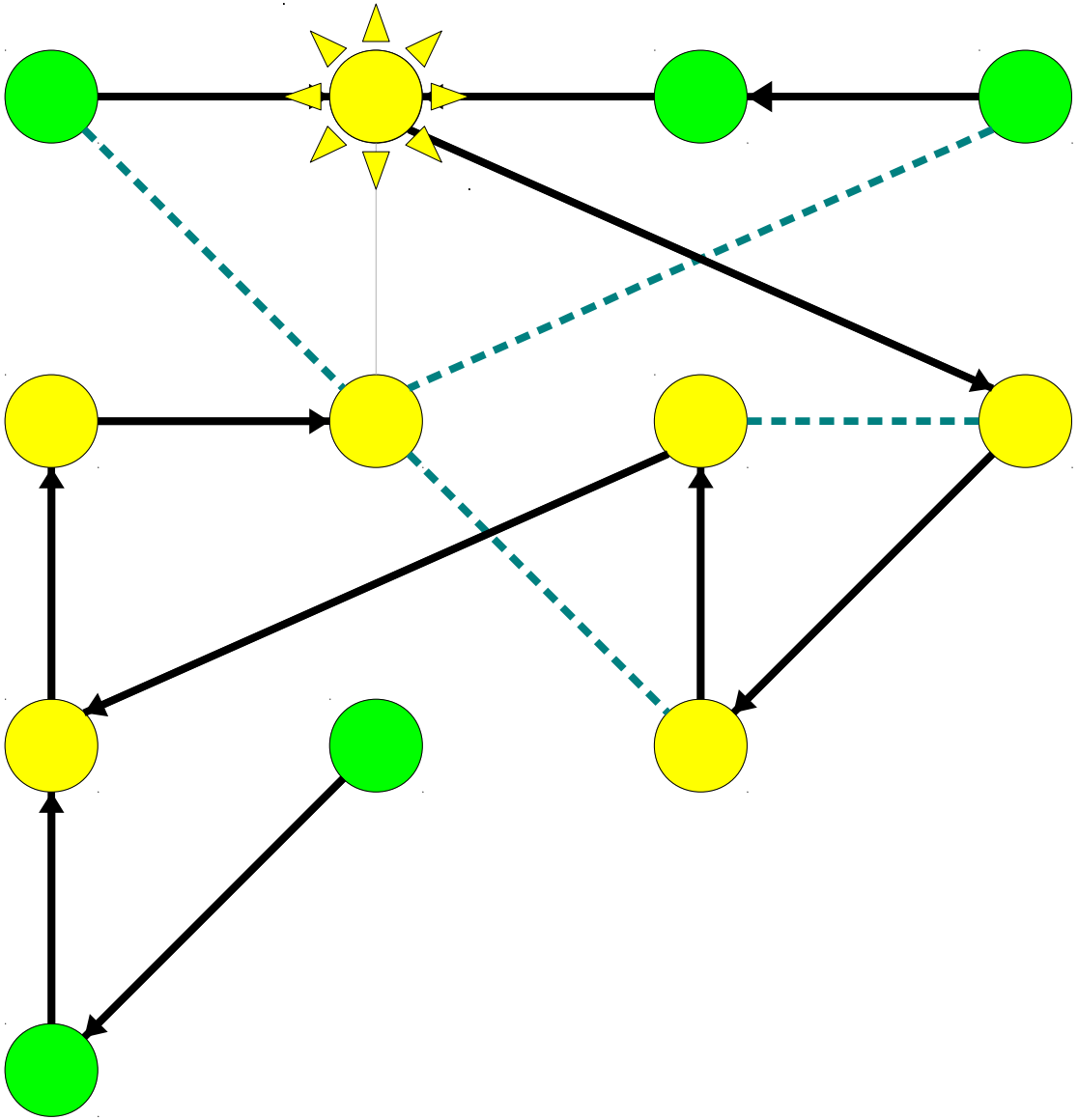
Depth-First Search



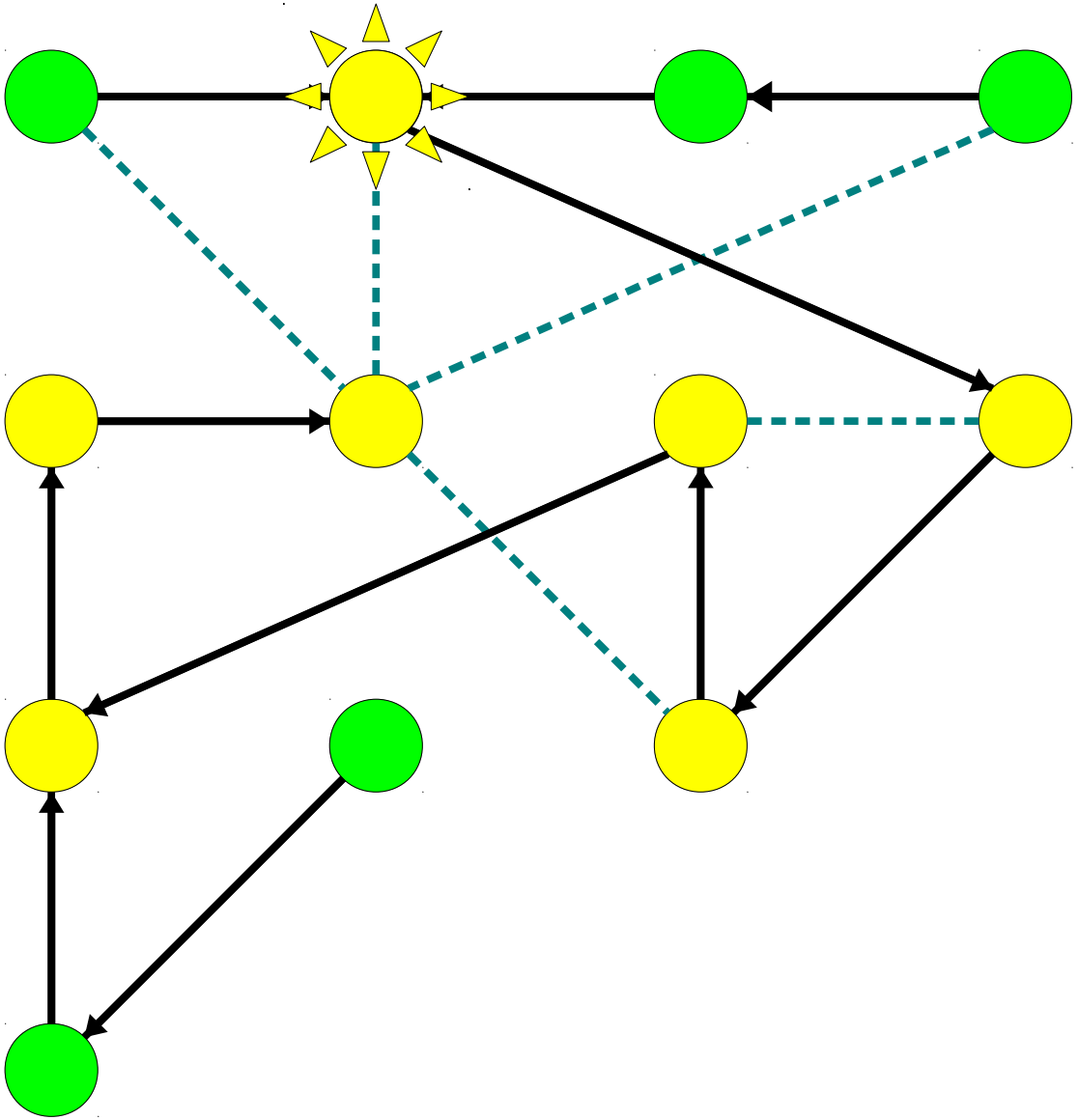
Depth-First Search



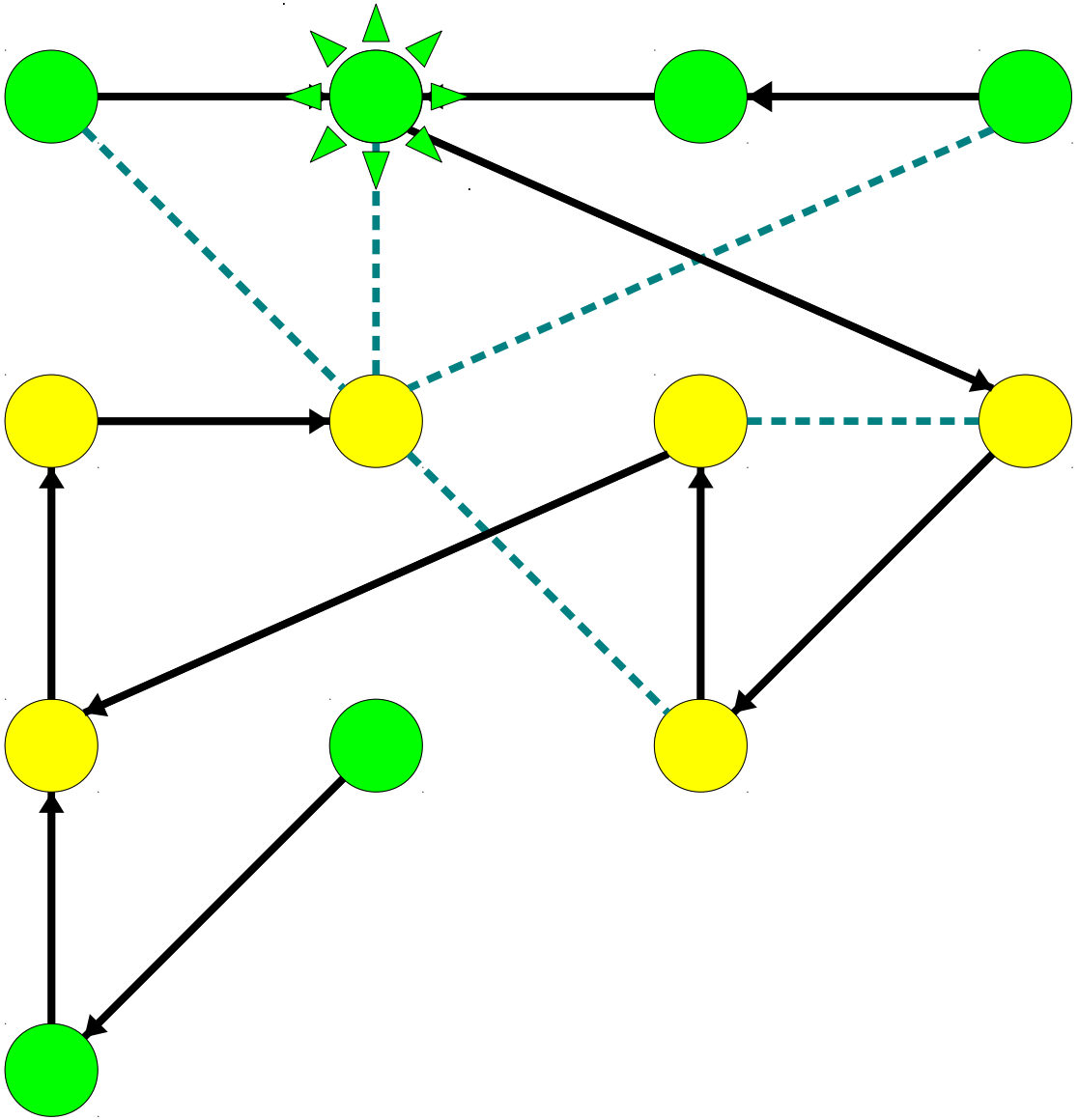
Depth-First Search



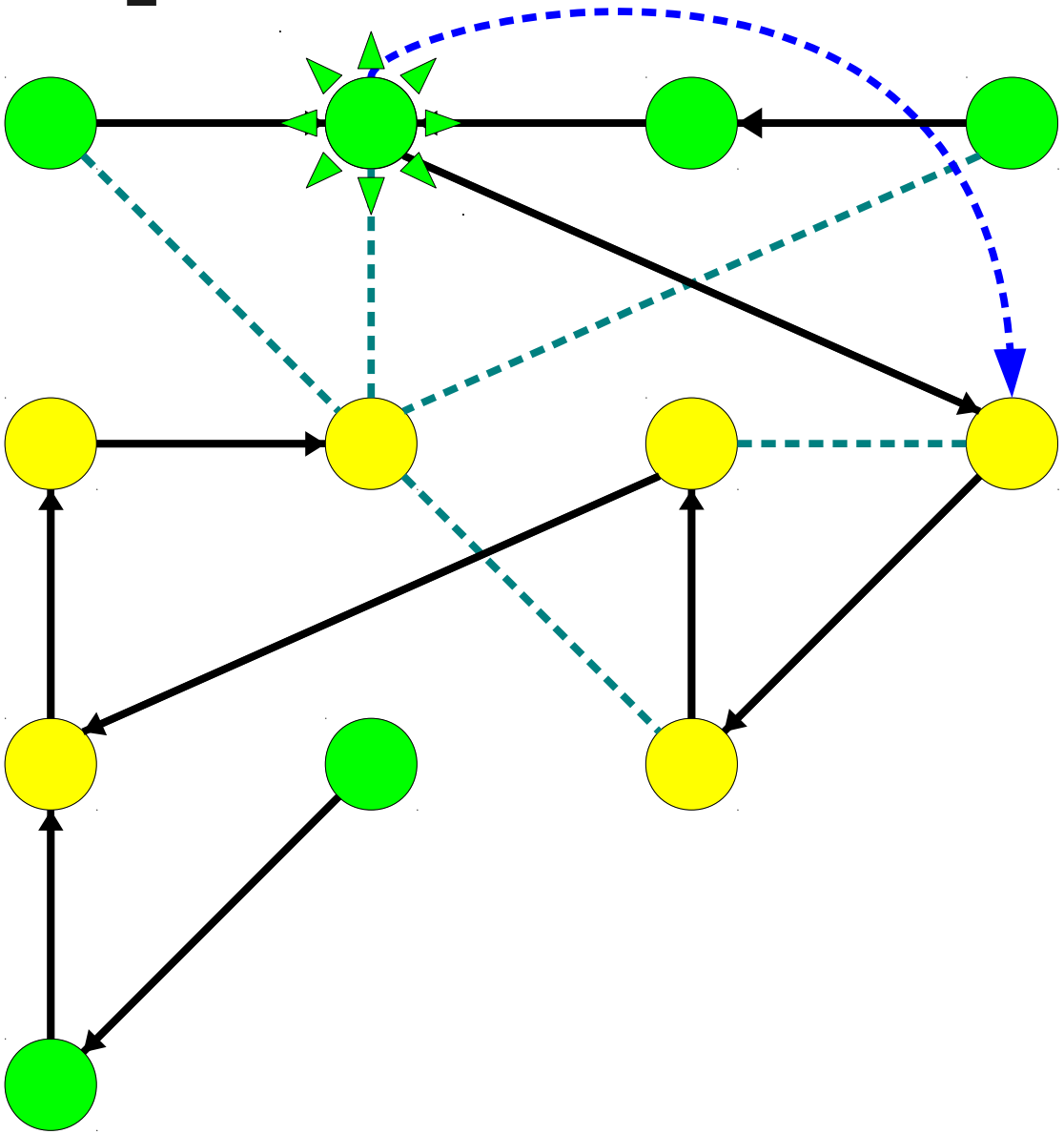
Depth-First Search



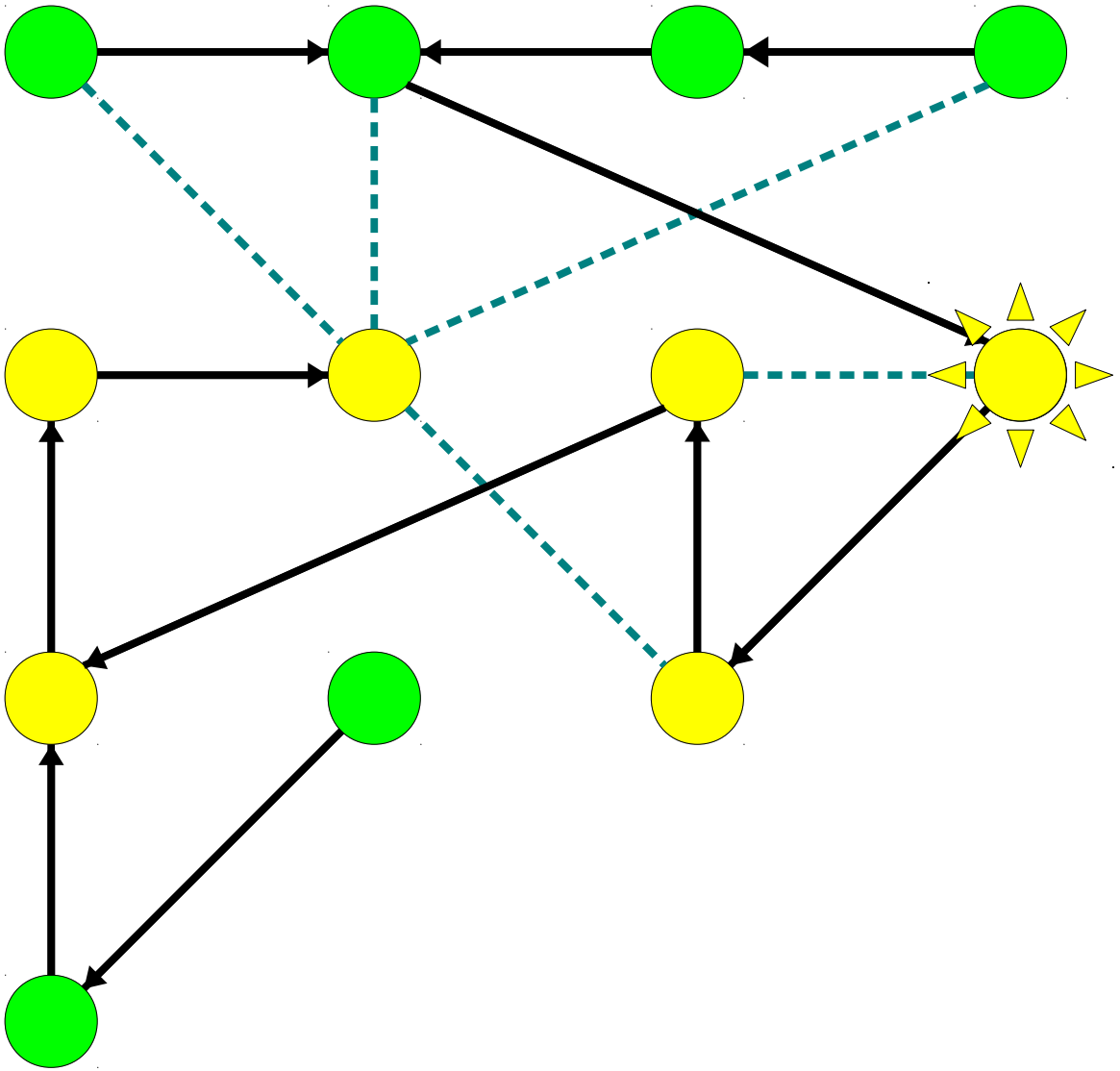
Depth-First Search



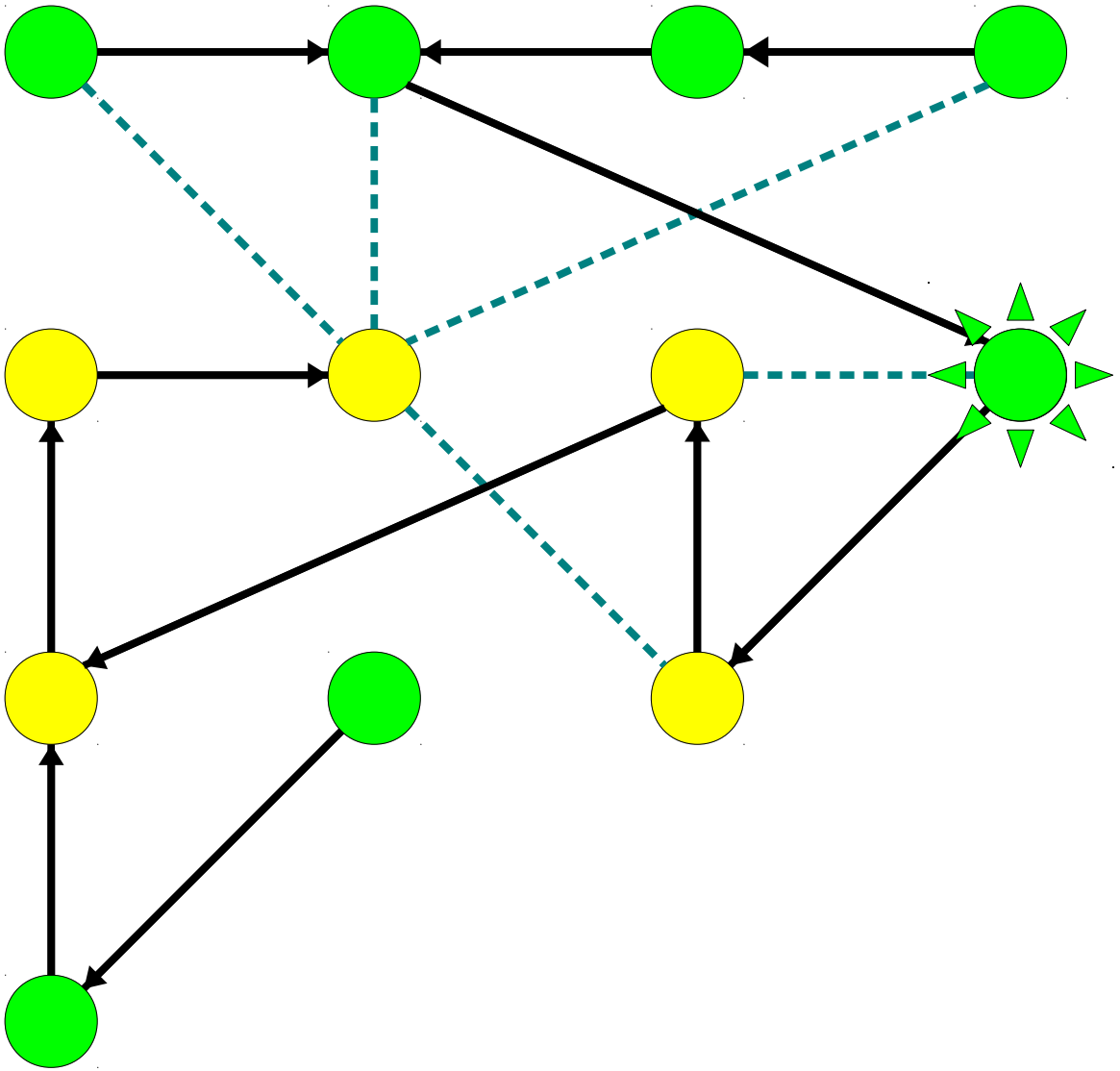
Depth-First Search



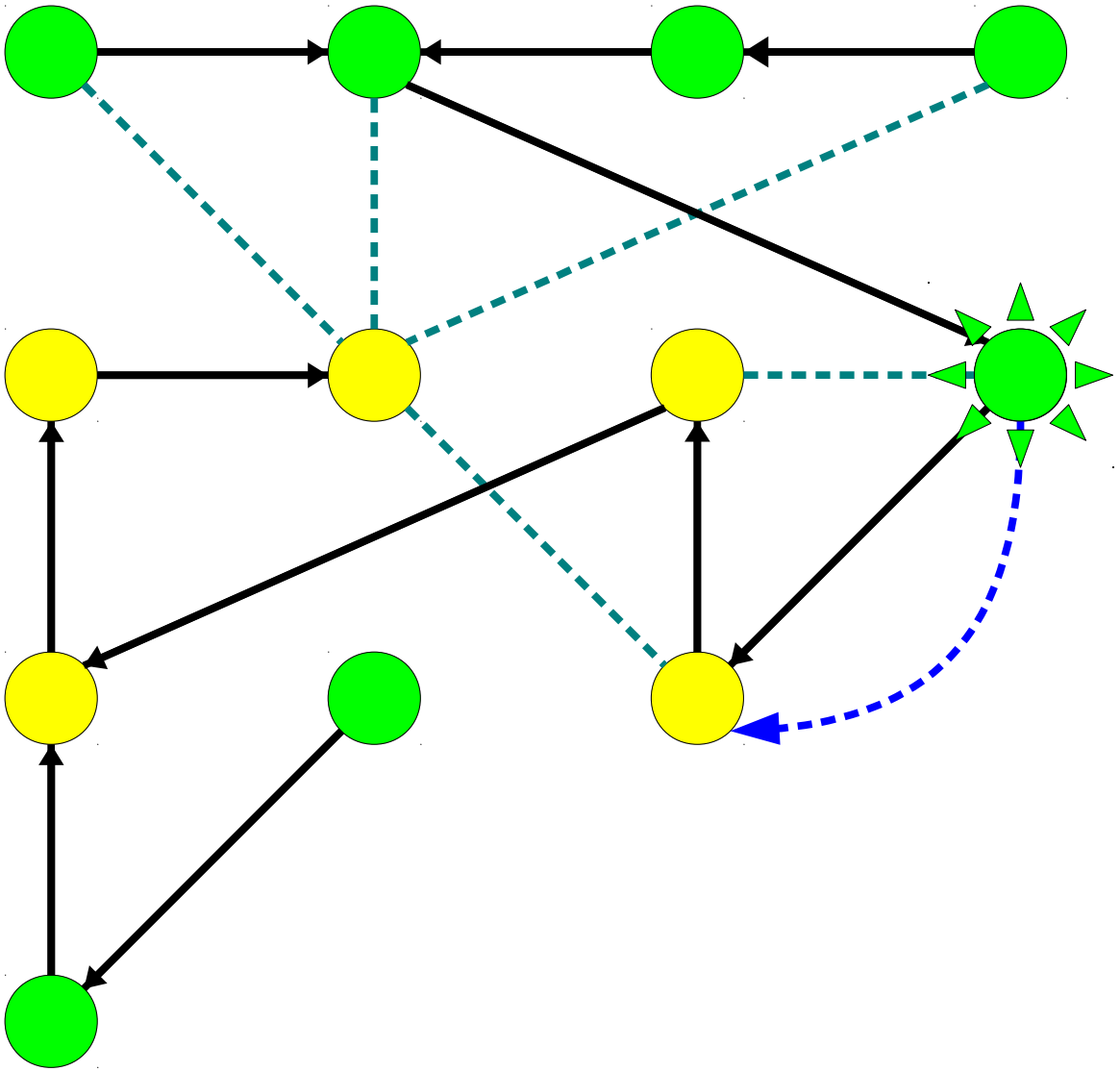
Depth-First Search



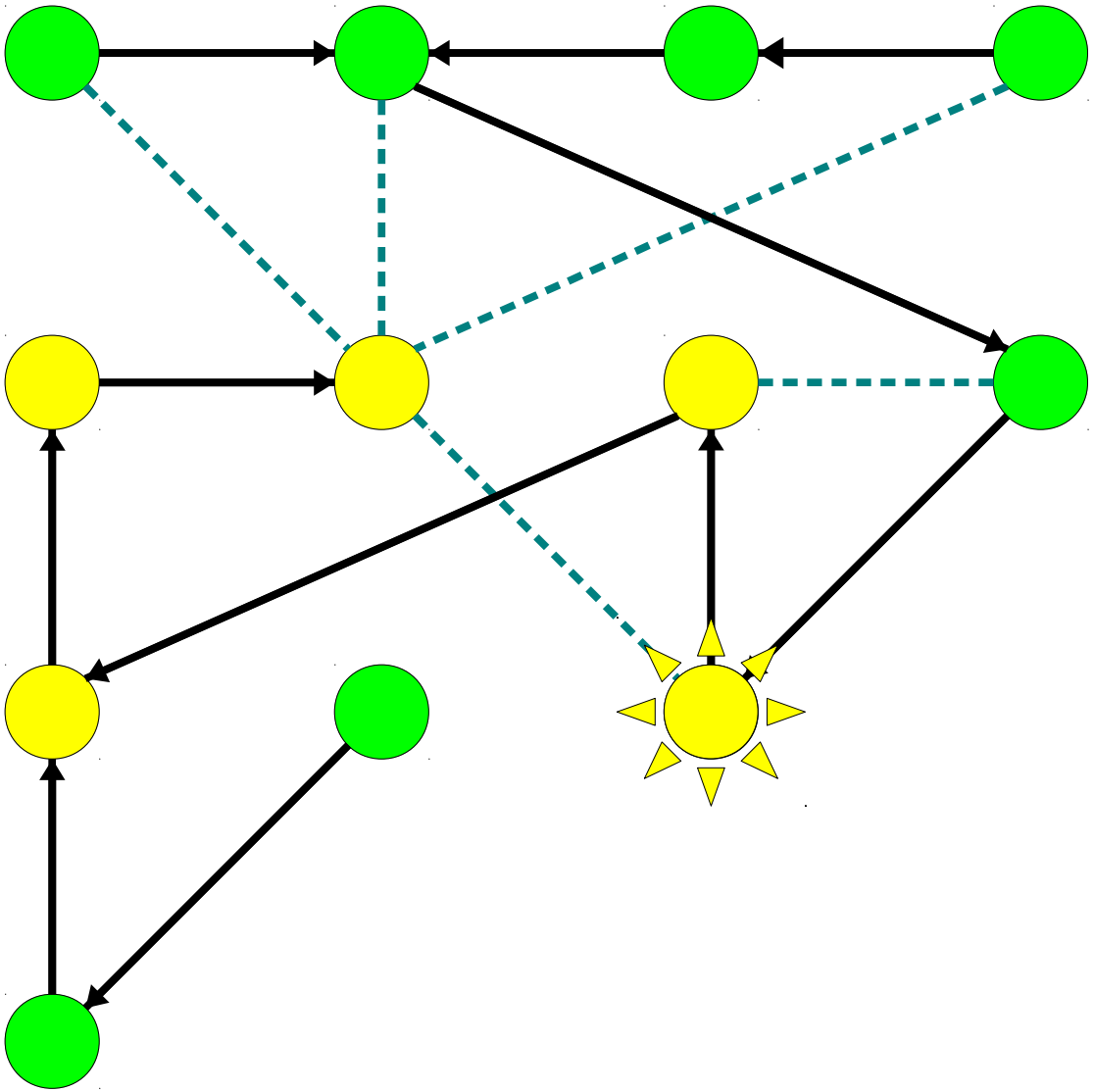
Depth-First Search



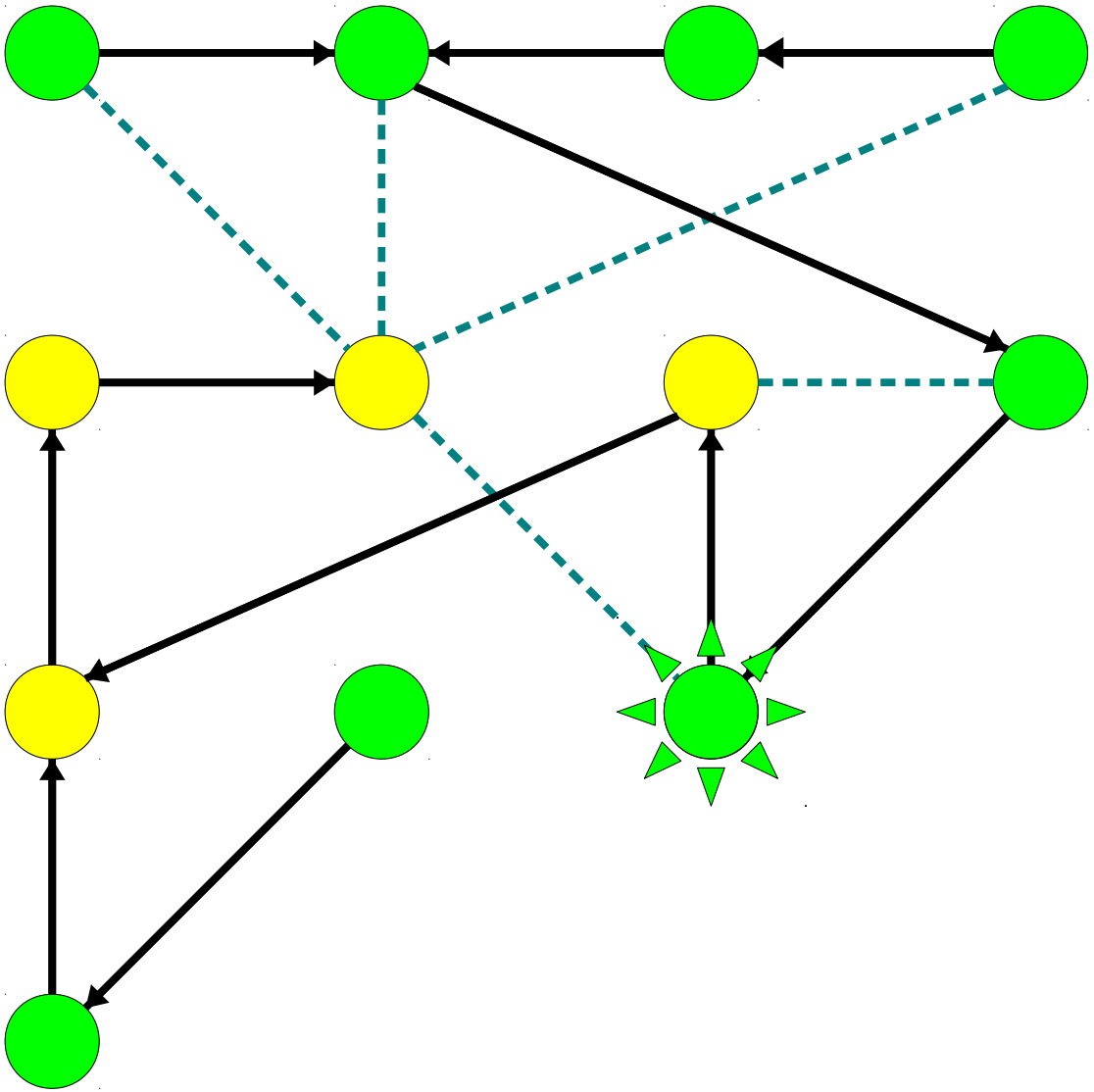
Depth-First Search



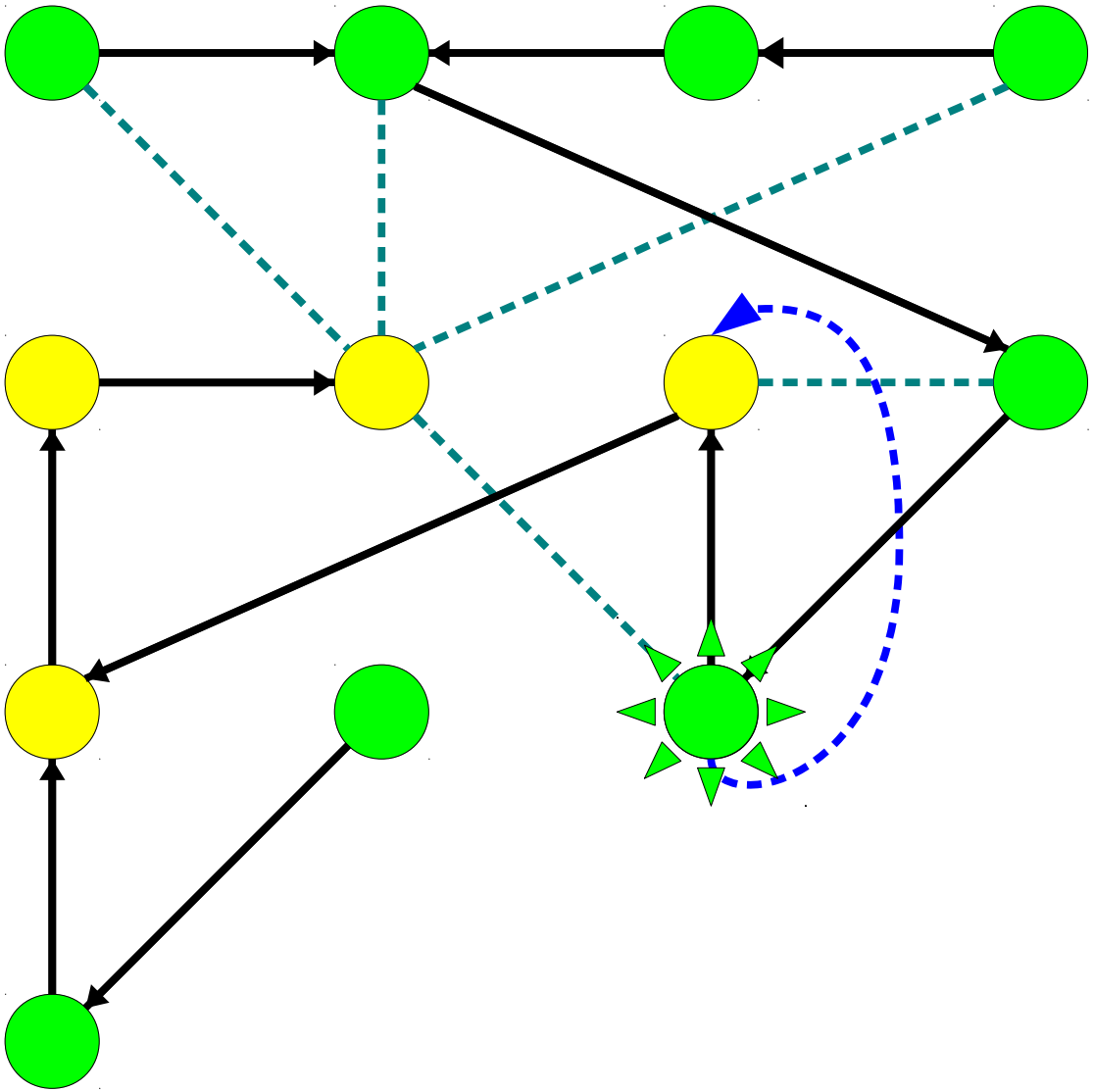
Depth-First Search



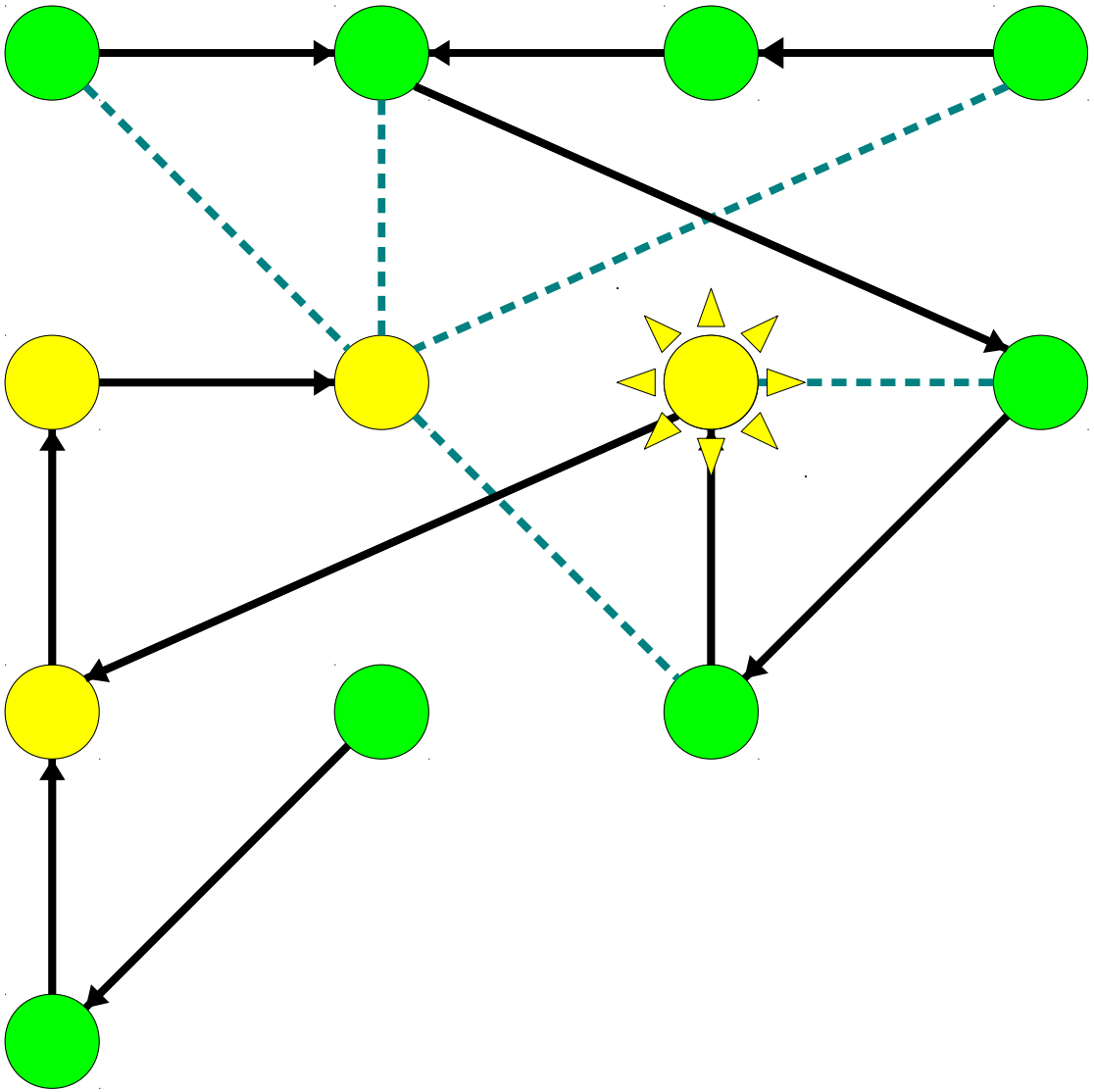
Depth-First Search



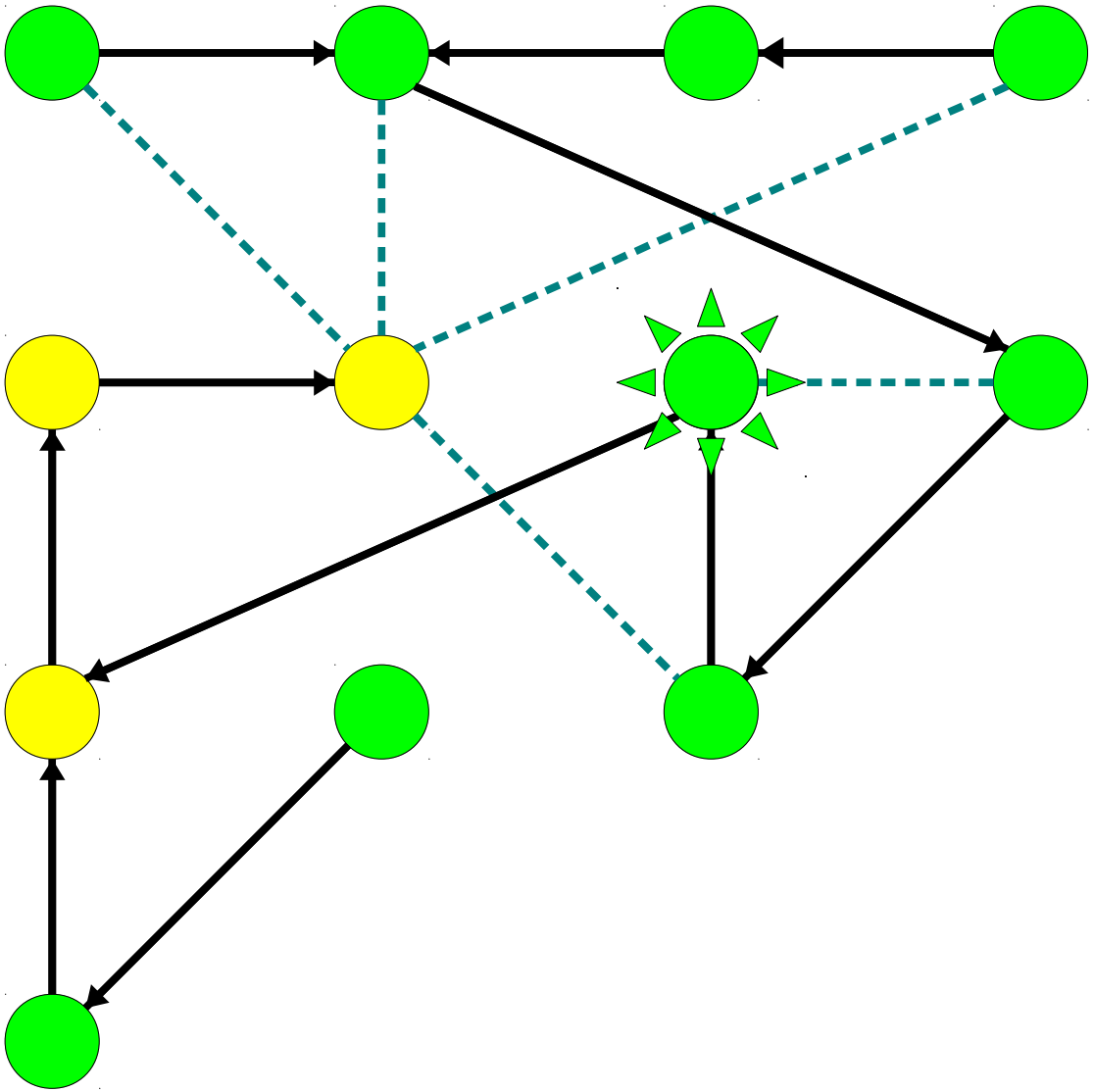
Depth-First Search



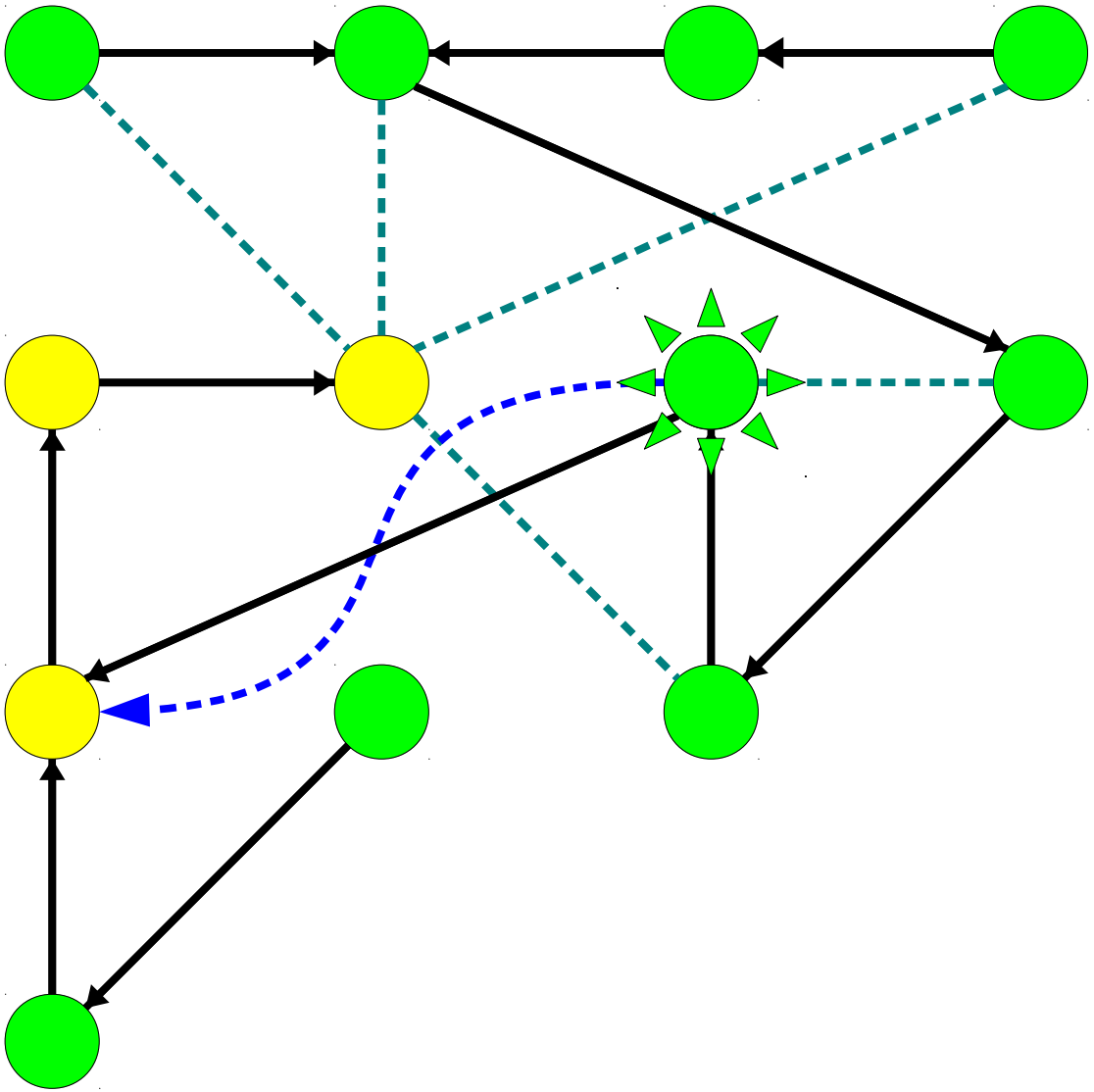
Depth-First Search



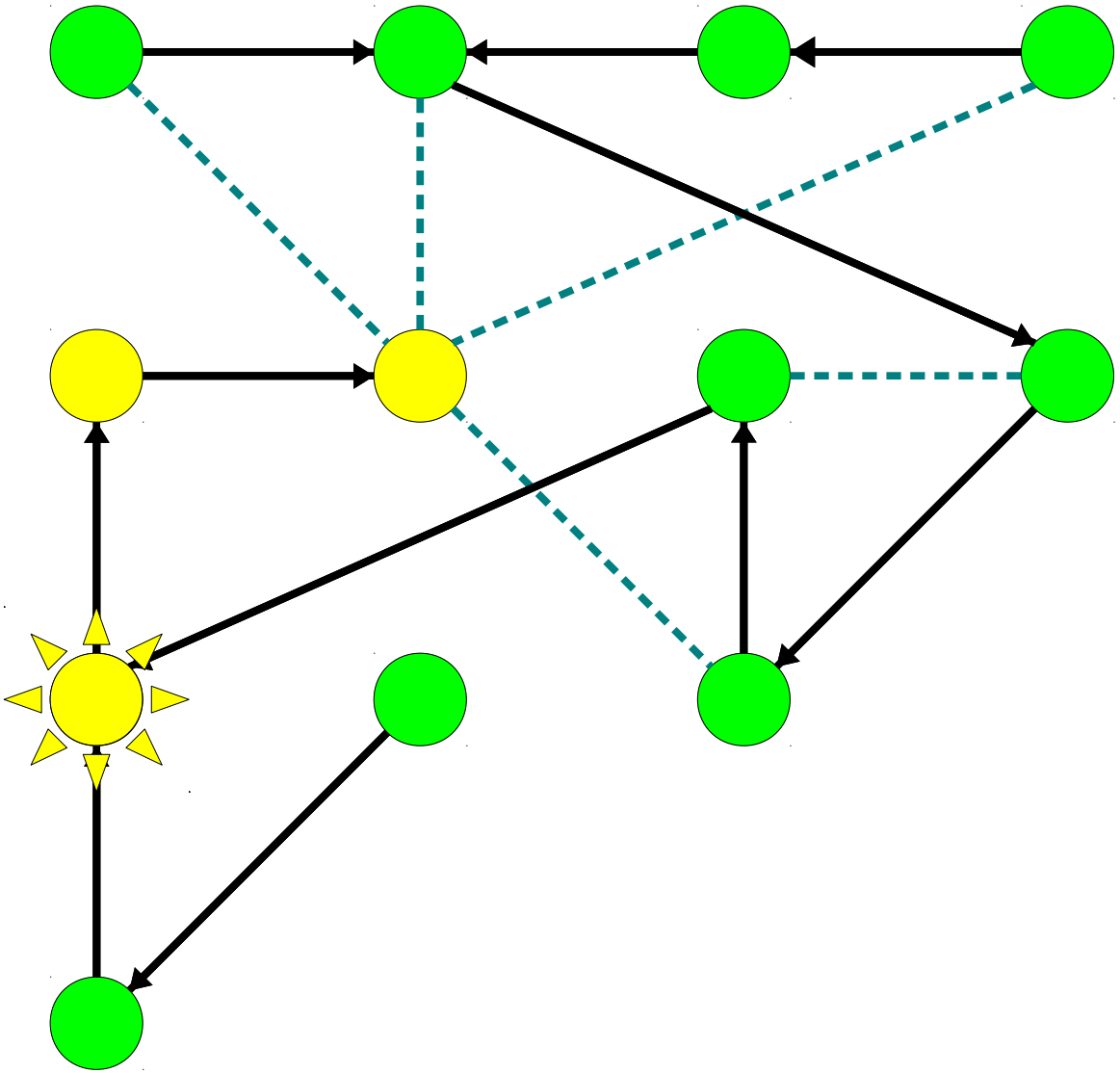
Depth-First Search



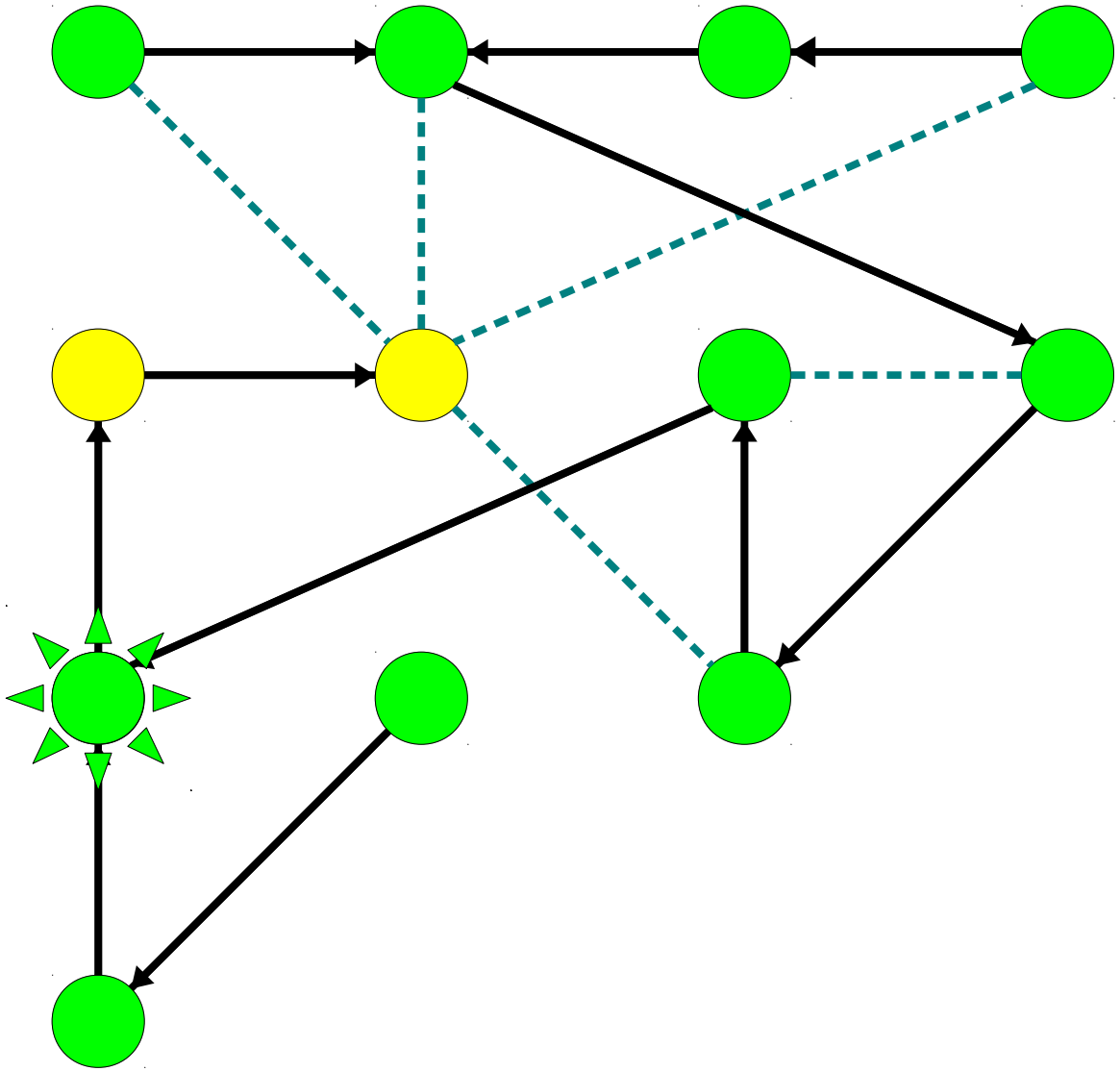
Depth-First Search



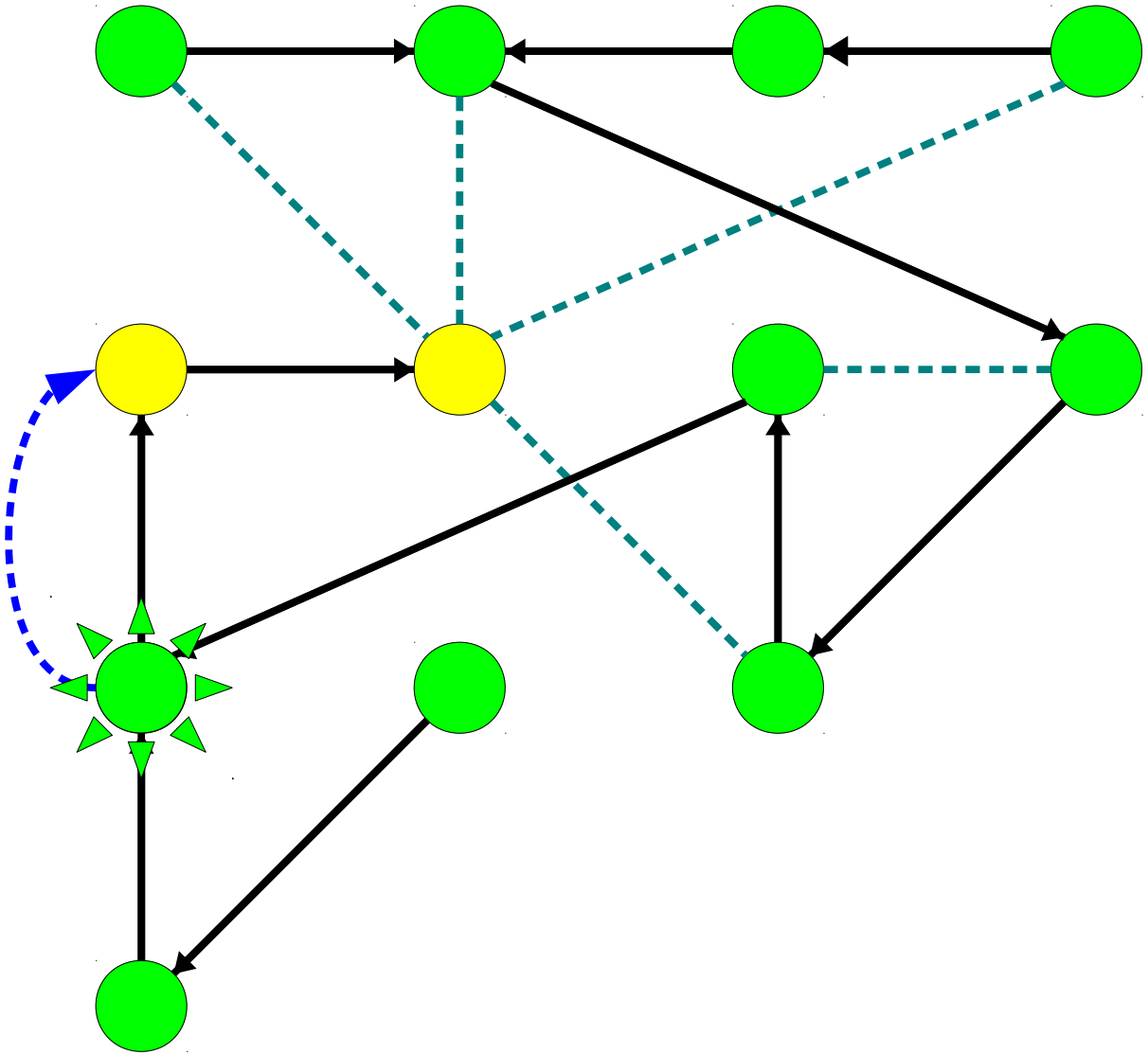
Depth-First Search



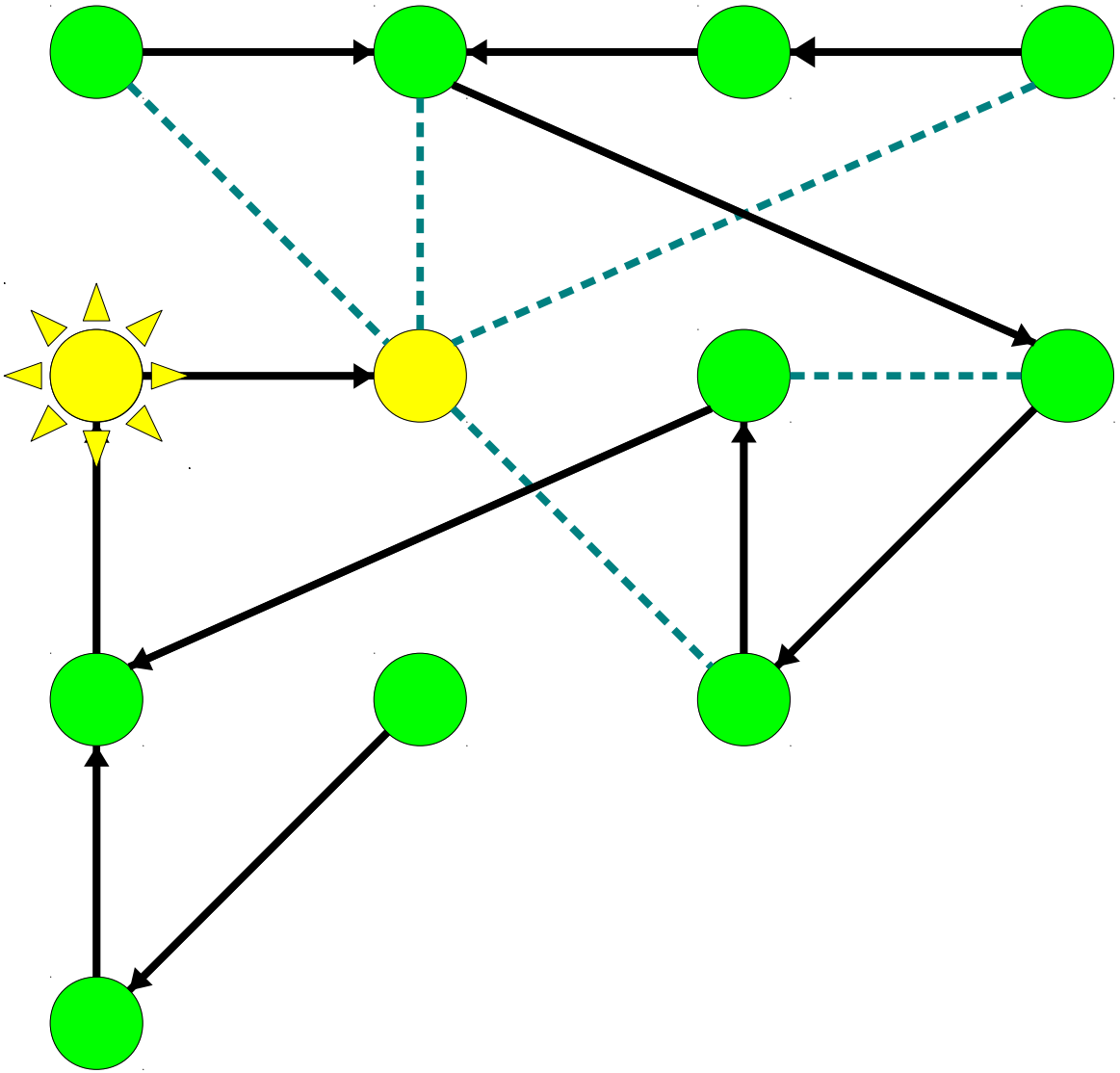
Depth-First Search



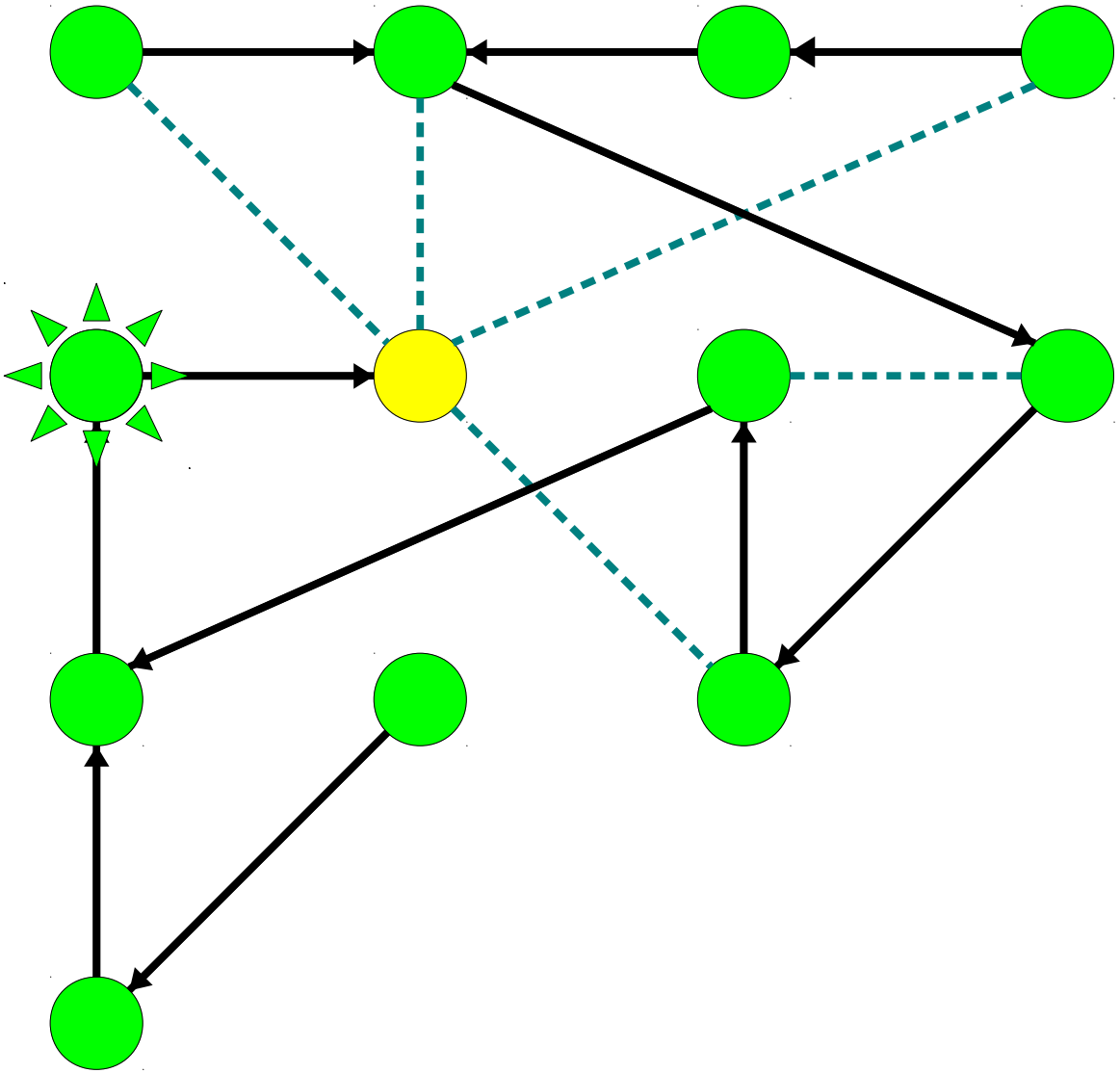
Depth-First Search



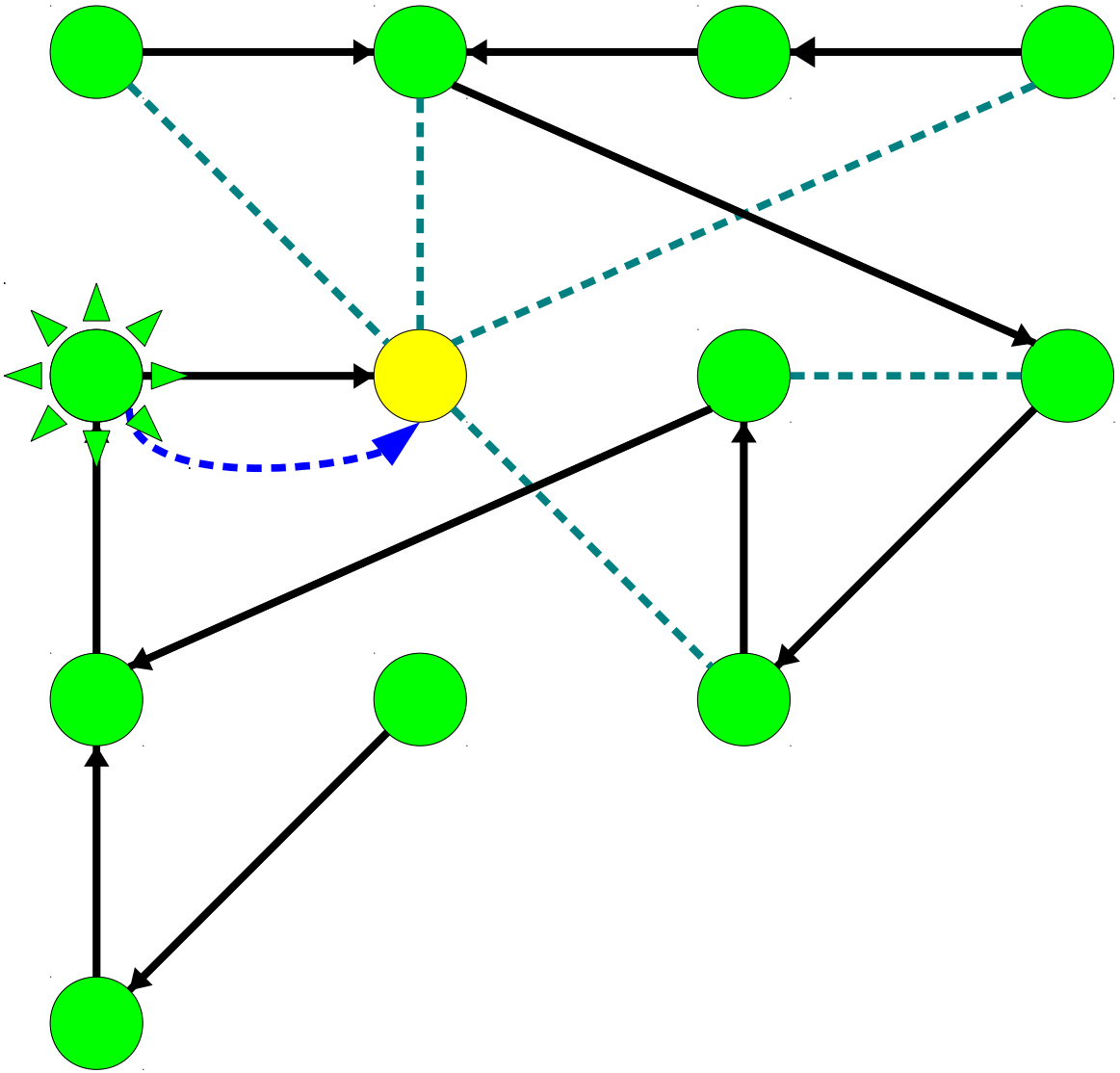
Depth-First Search



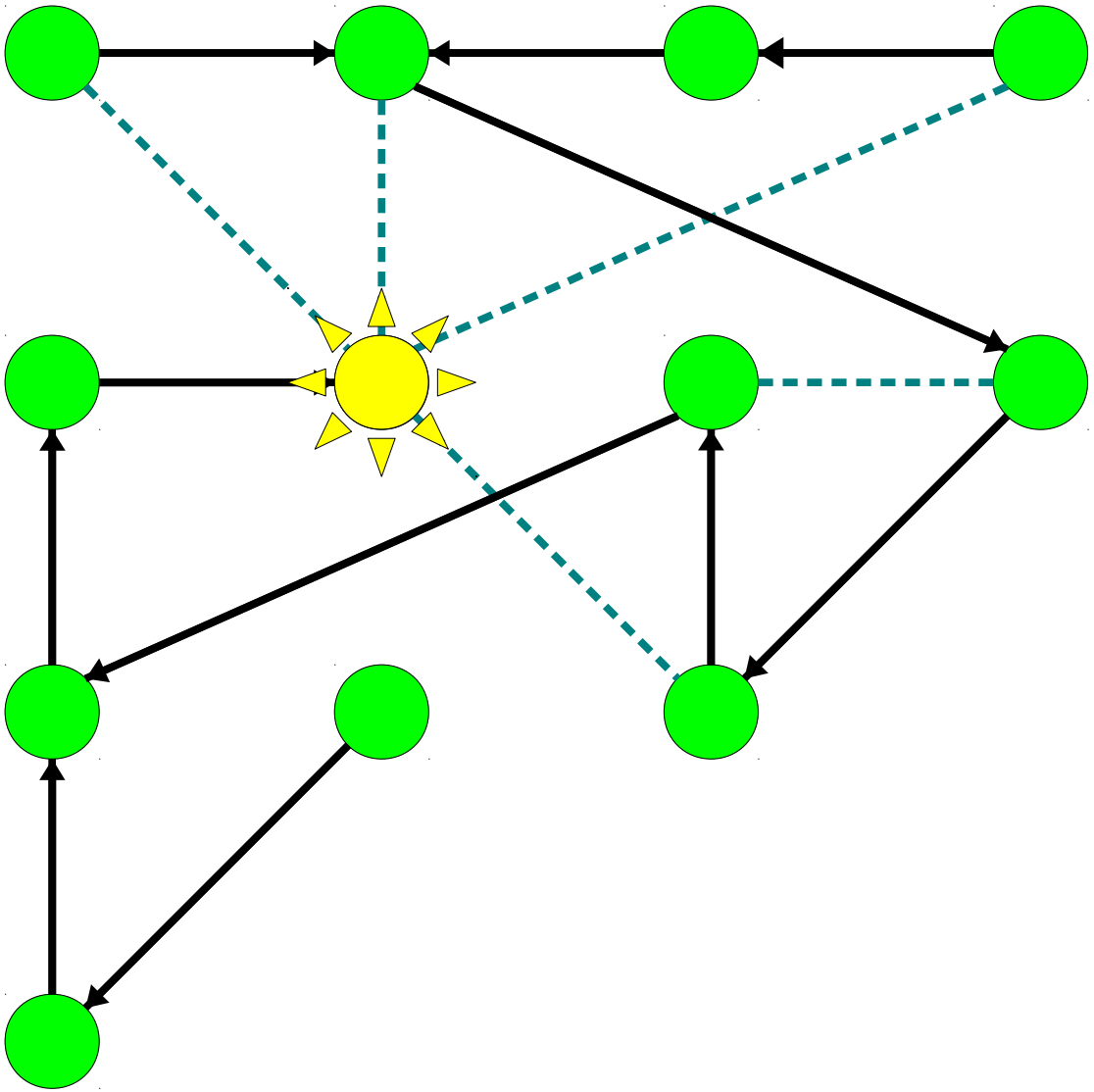
Depth-First Search



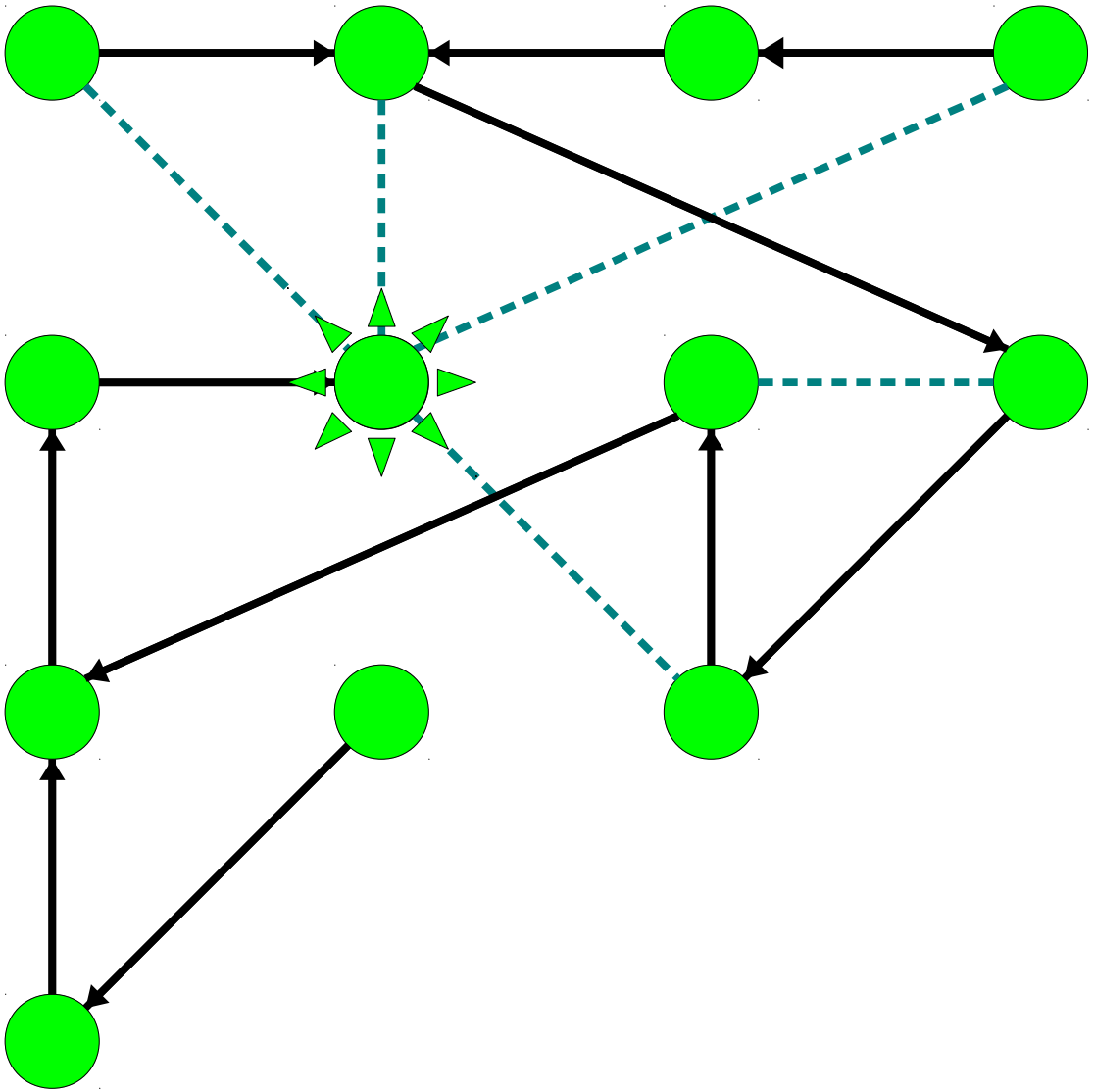
Depth-First Search



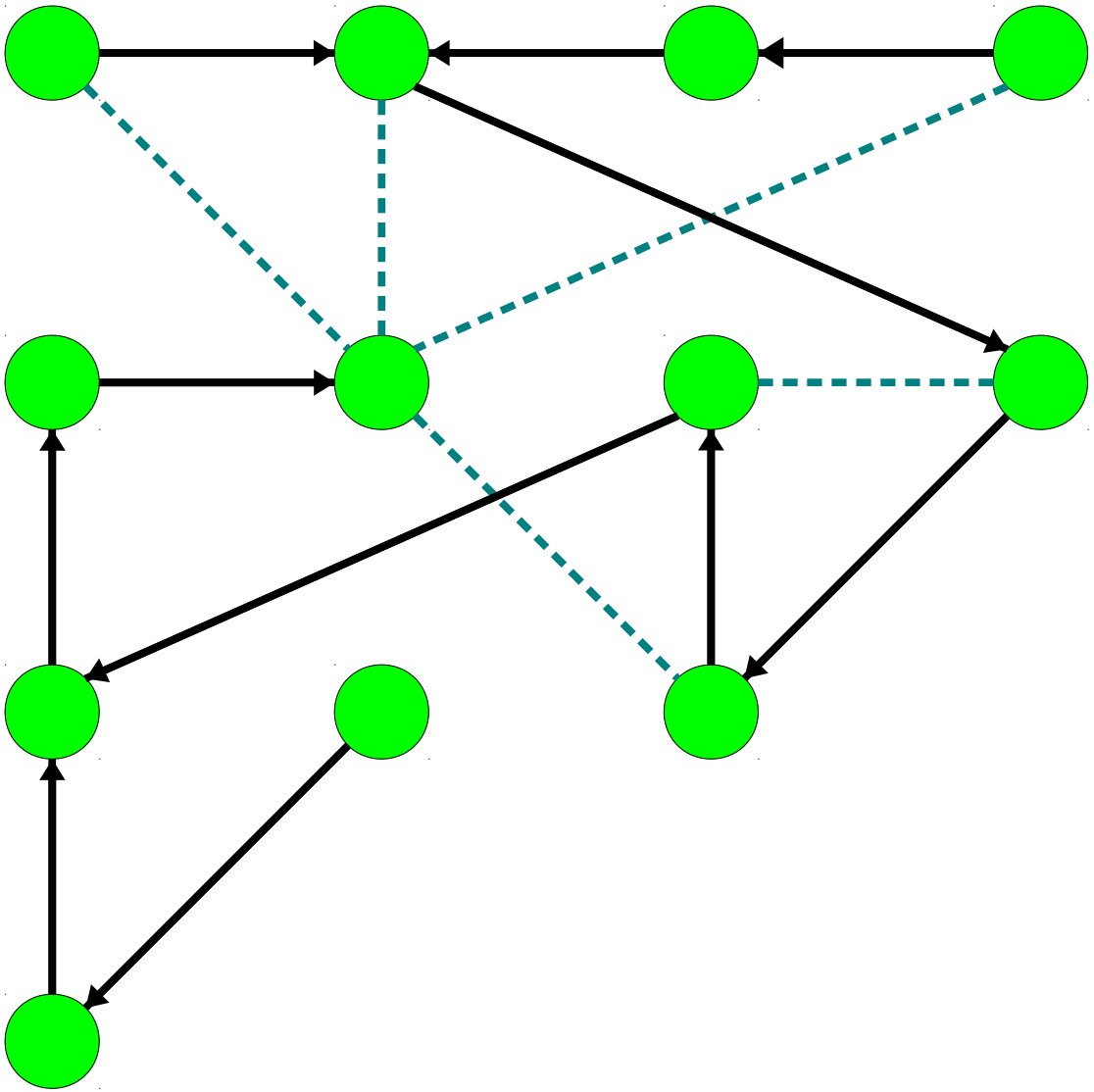
Depth-First Search



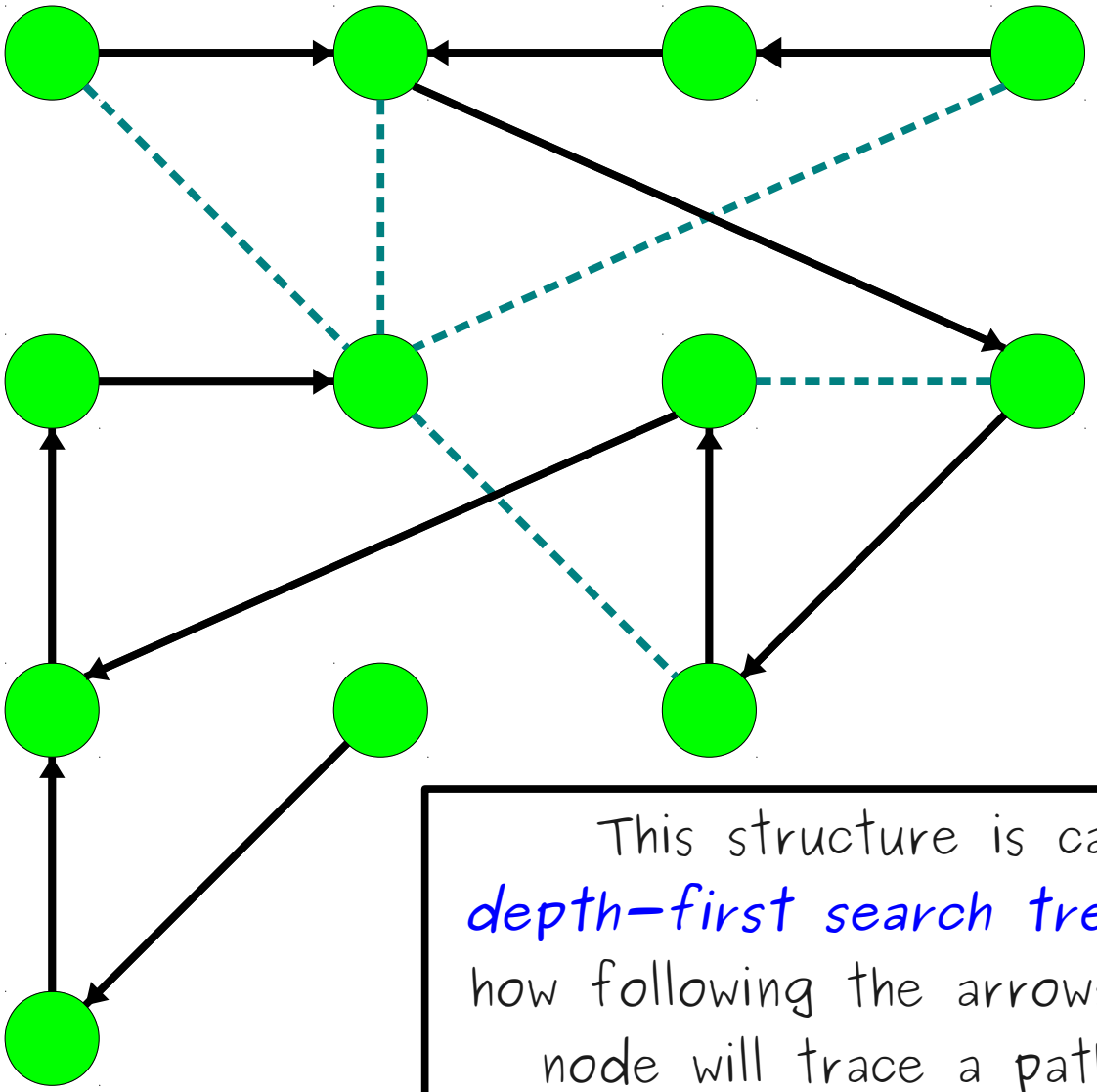
Depth-First Search



Depth-First Search

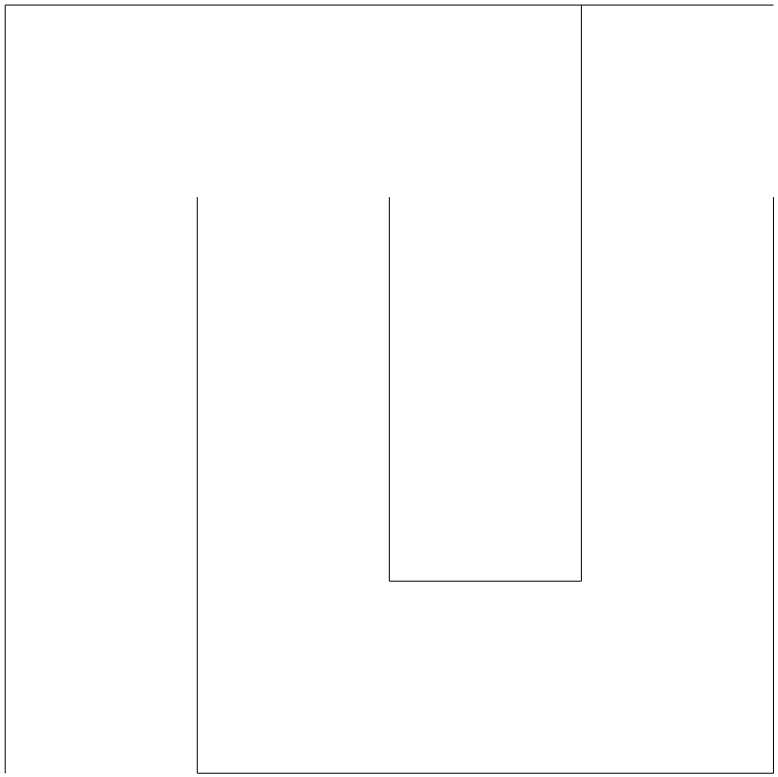


Depth-First Search

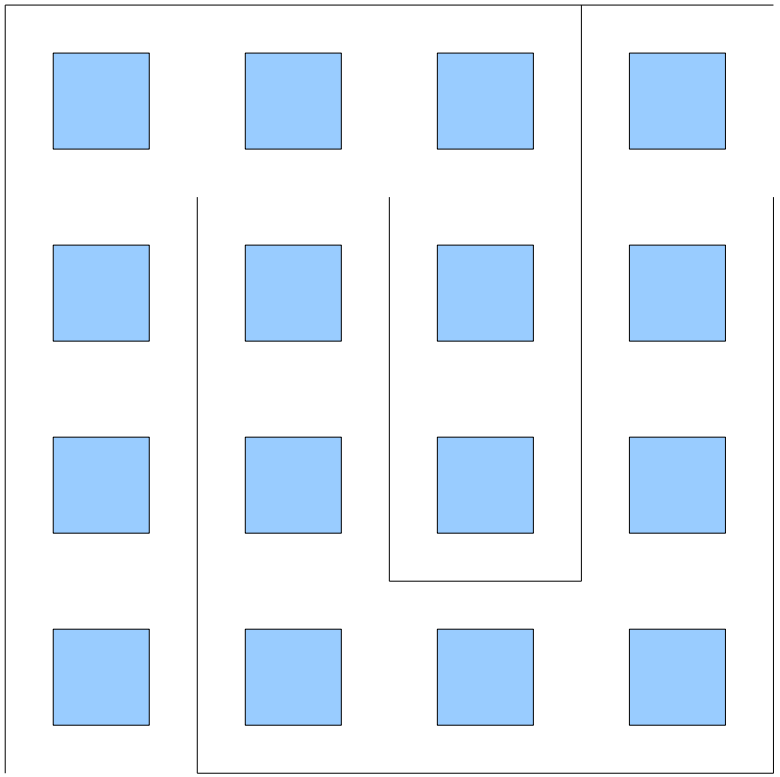


This structure is called a *depth-first search tree*. Notice how following the arrows from any node will trace a path to the starting node in reverse.

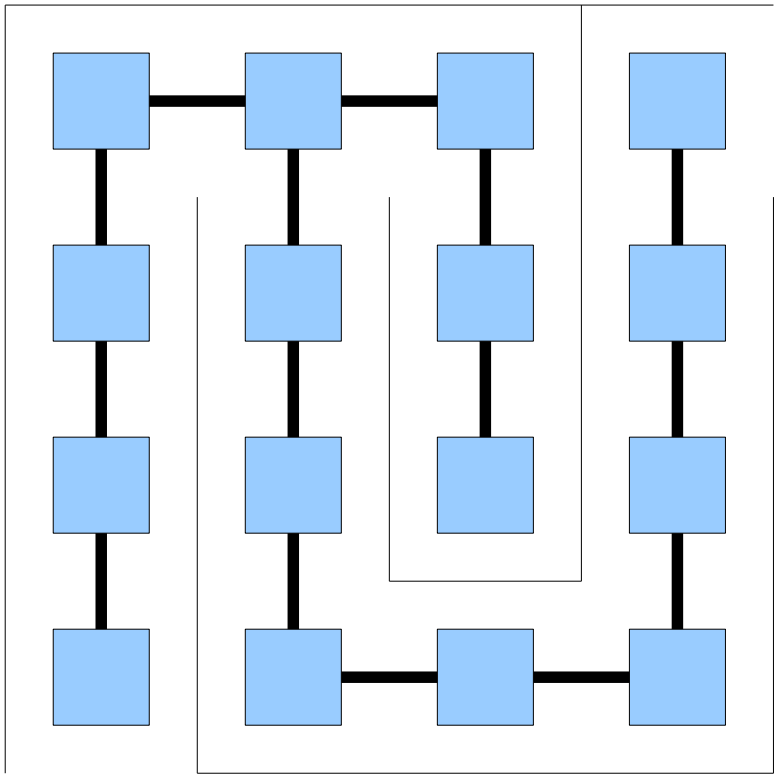
Mazes as Graphs



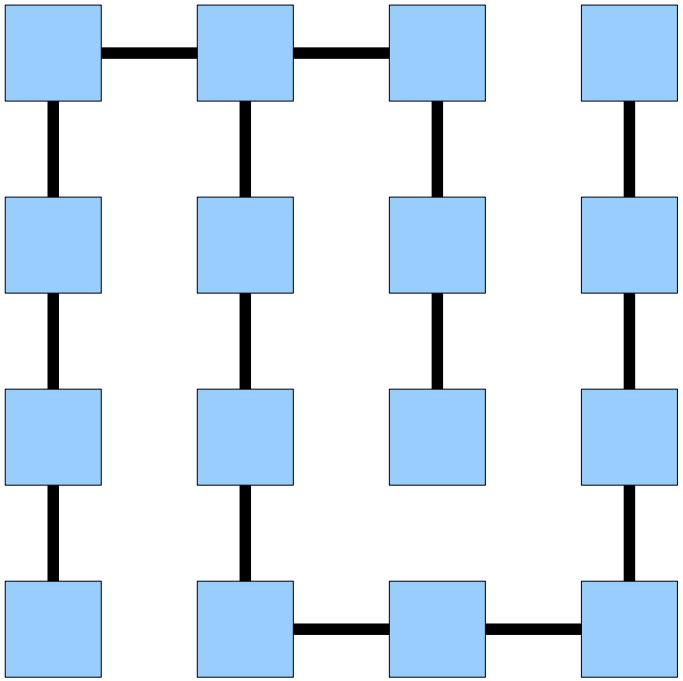
Mazes as Graphs



Mazes as Graphs

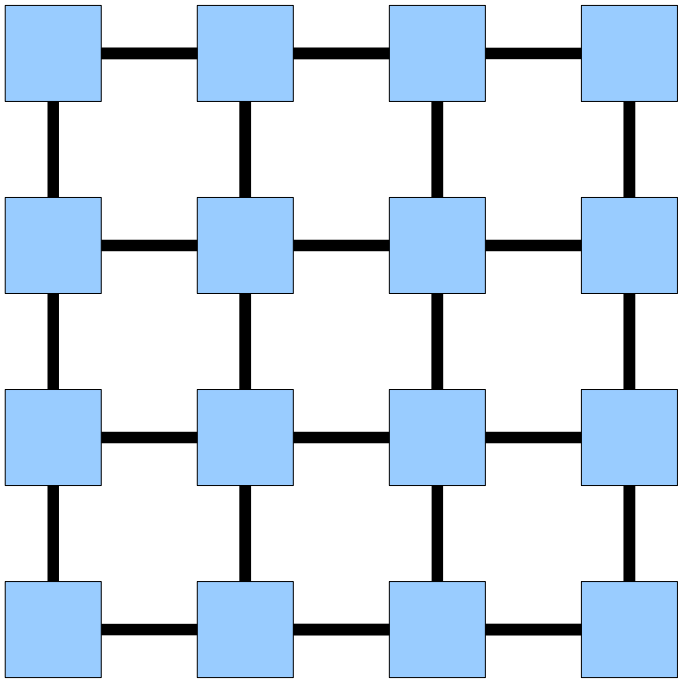


Mazes as Graphs



Creating a Maze with DFS

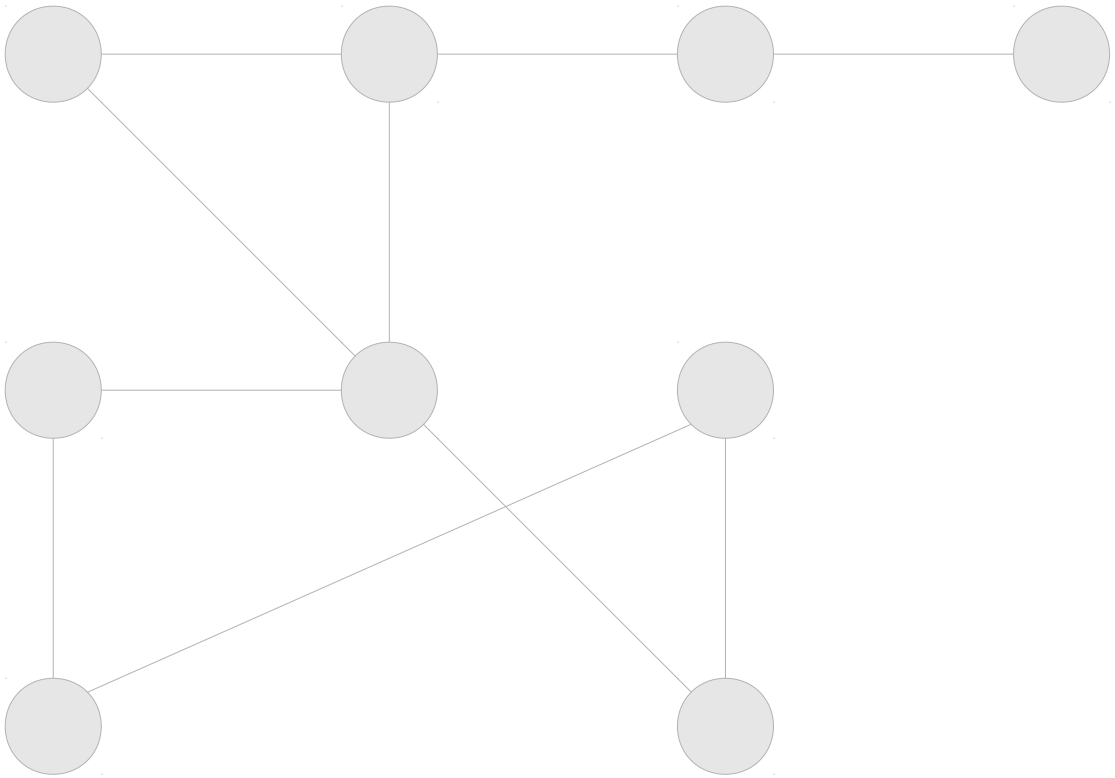
- Create a **grid graph** of the appropriate size.



- Starting at any node, run a depth-first search, choosing neighbor orderings at random.
- The resulting DFS tree is a maze with one solution.

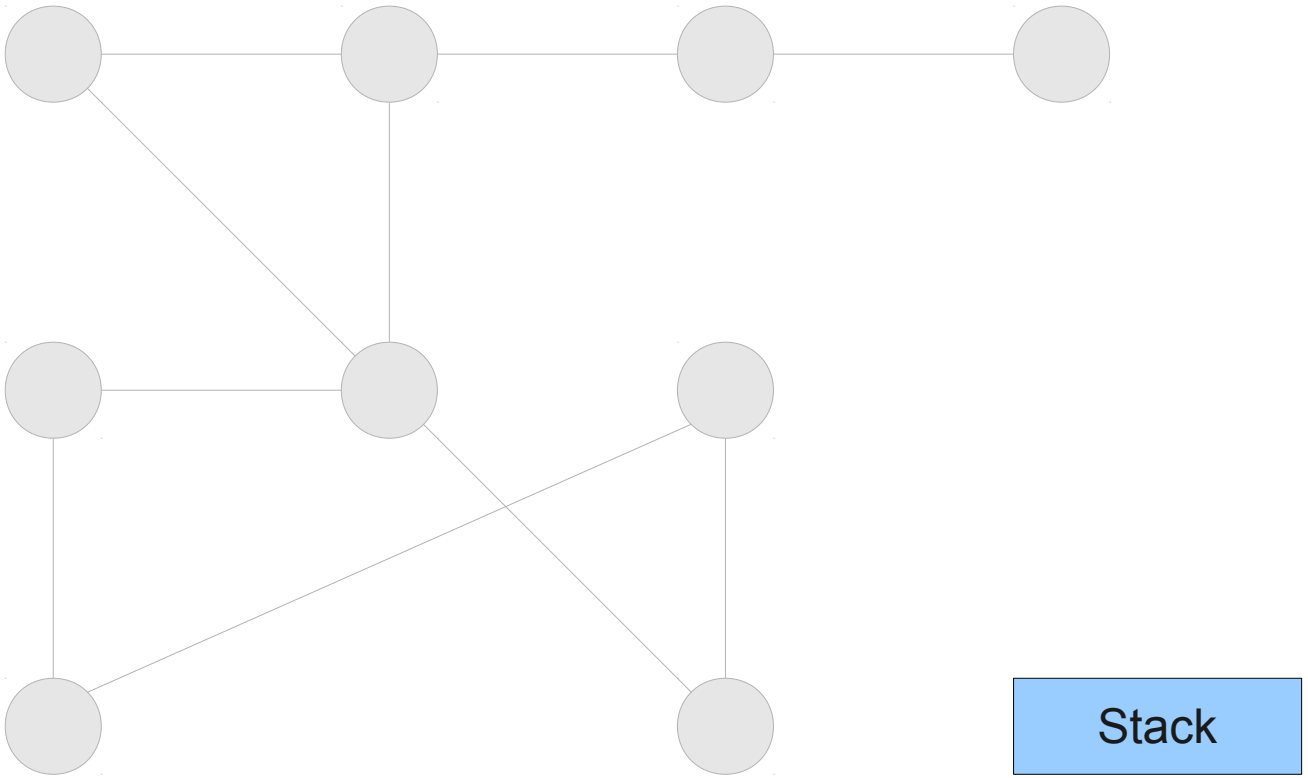
Problems with DFS

- What happens if you want to search for a specific node in the graph?



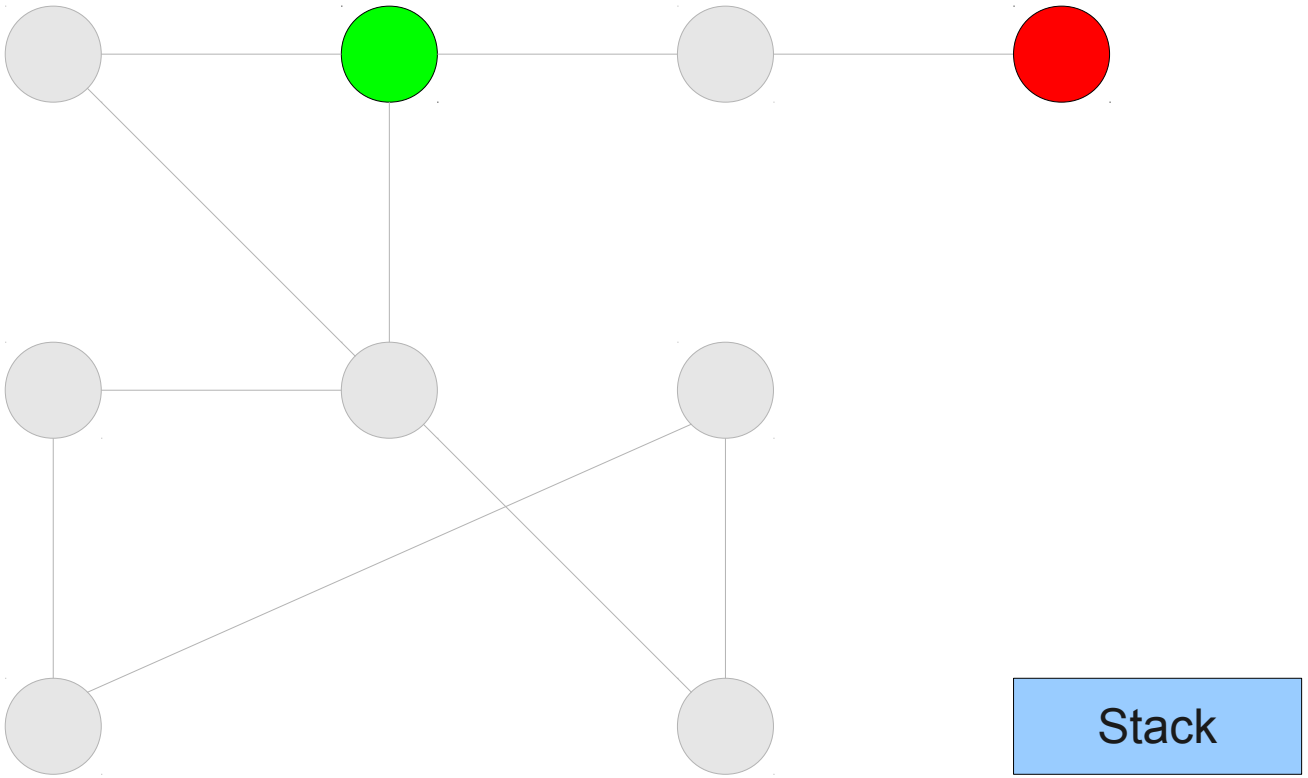
Problems with DFS

- What happens if you want to search for a specific node in the graph?



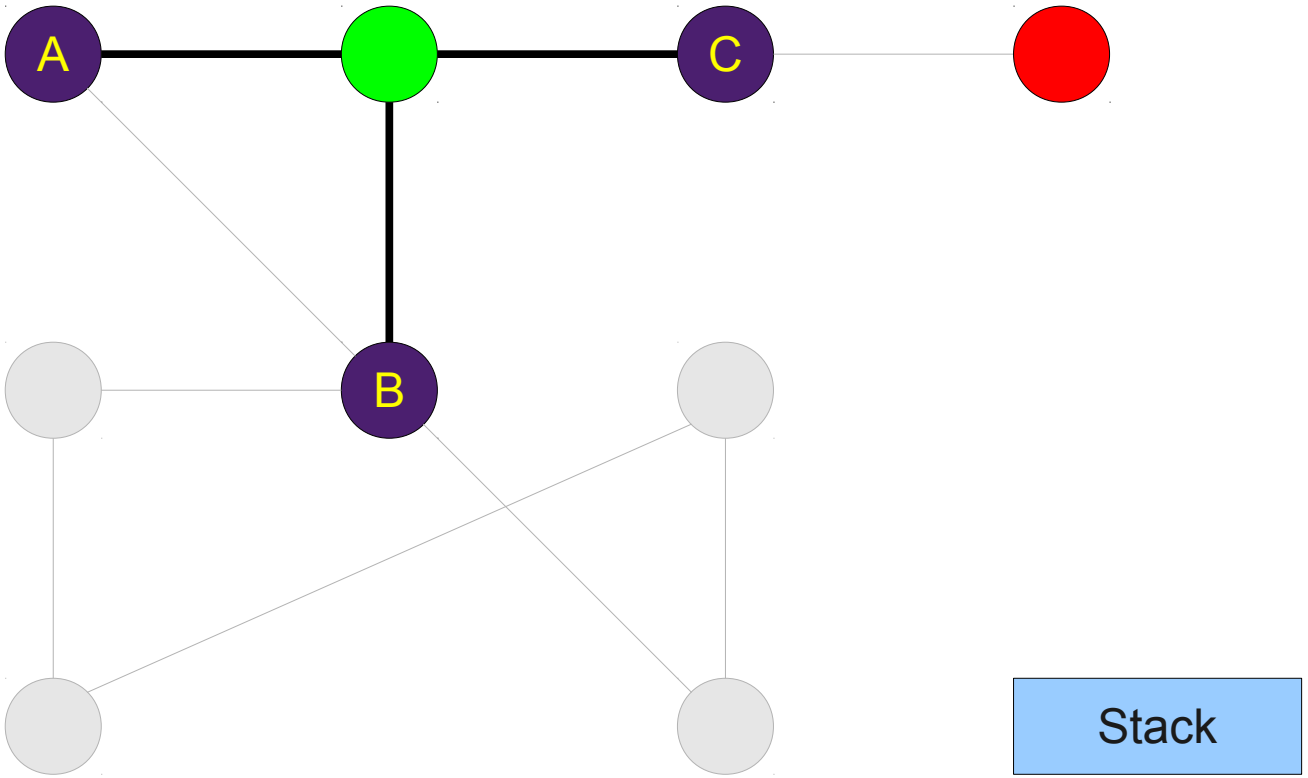
Problems with DFS

- What happens if you want to search for a specific node in the graph?



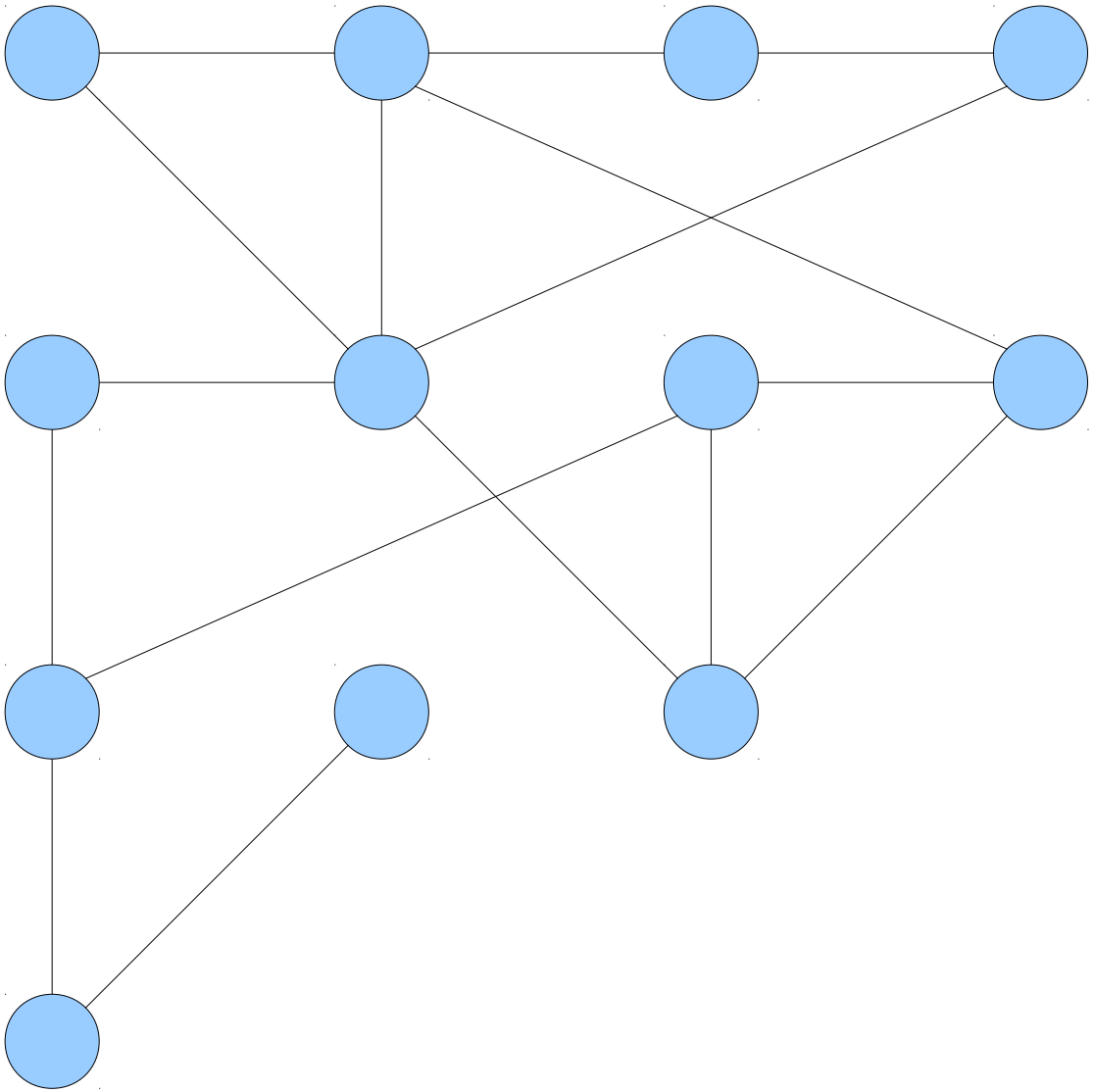
Problems with DFS

- What happens if you want to search for a specific node in the graph?

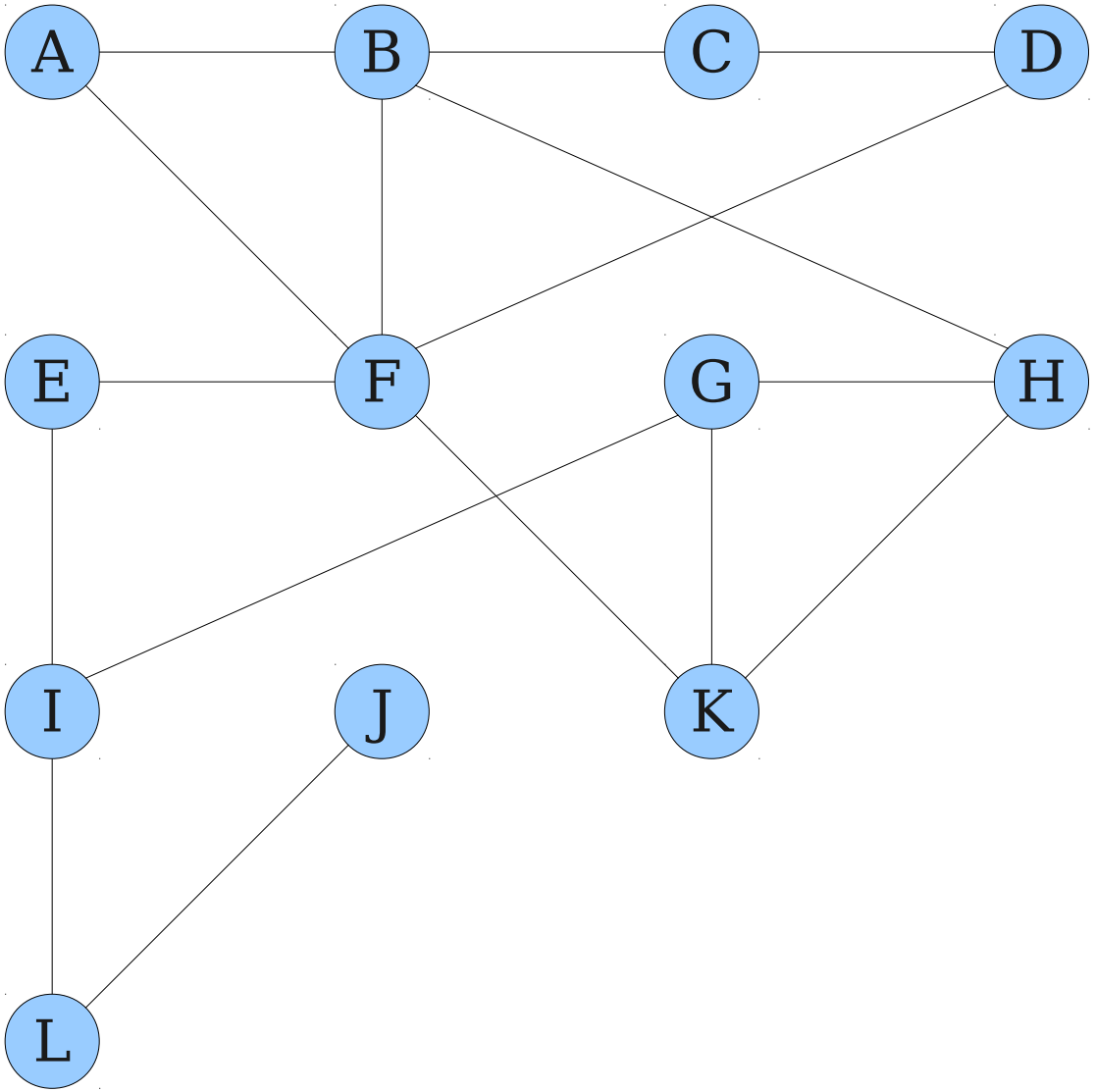


Breadth-First Search

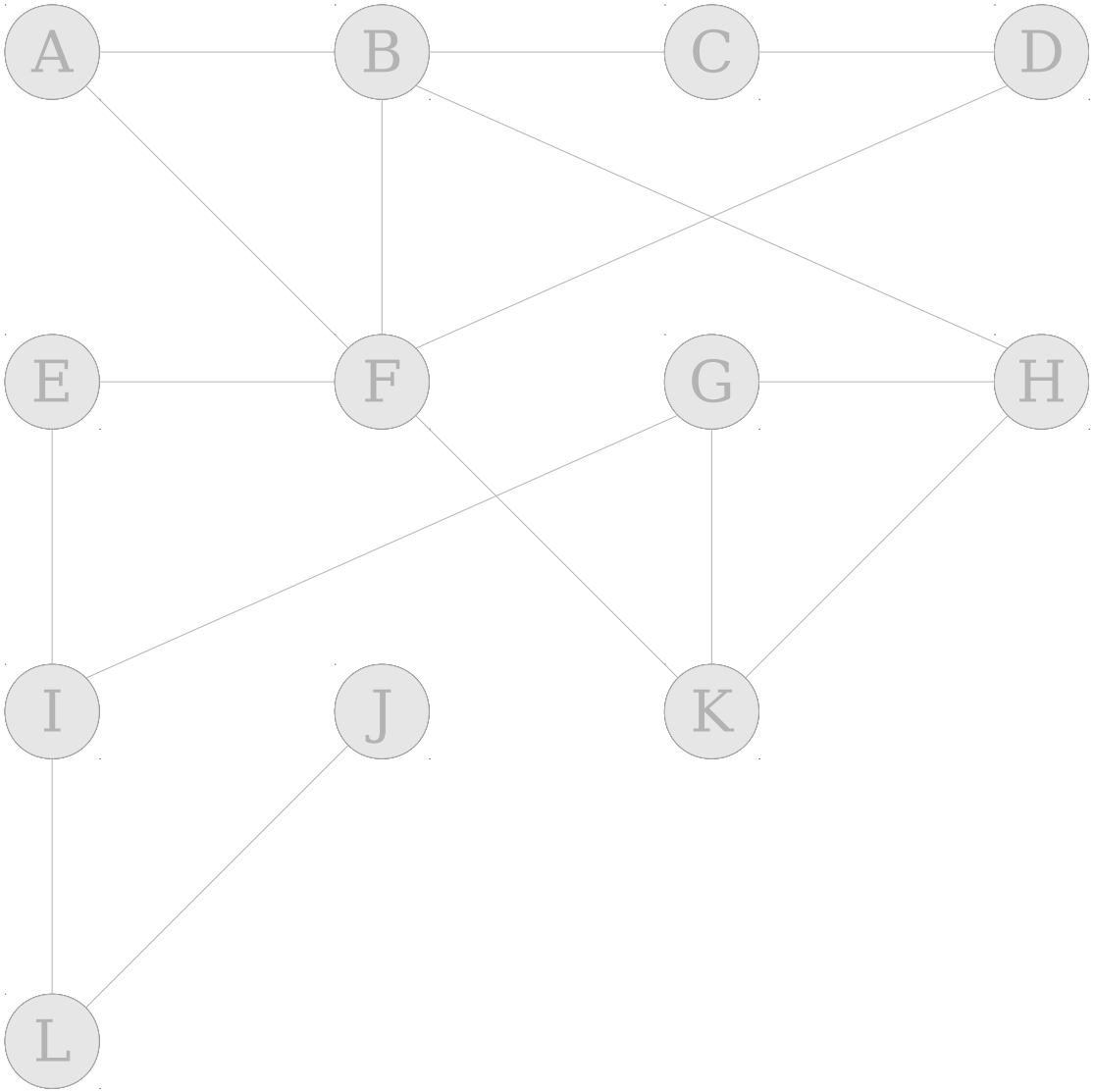
Breadth-First Search



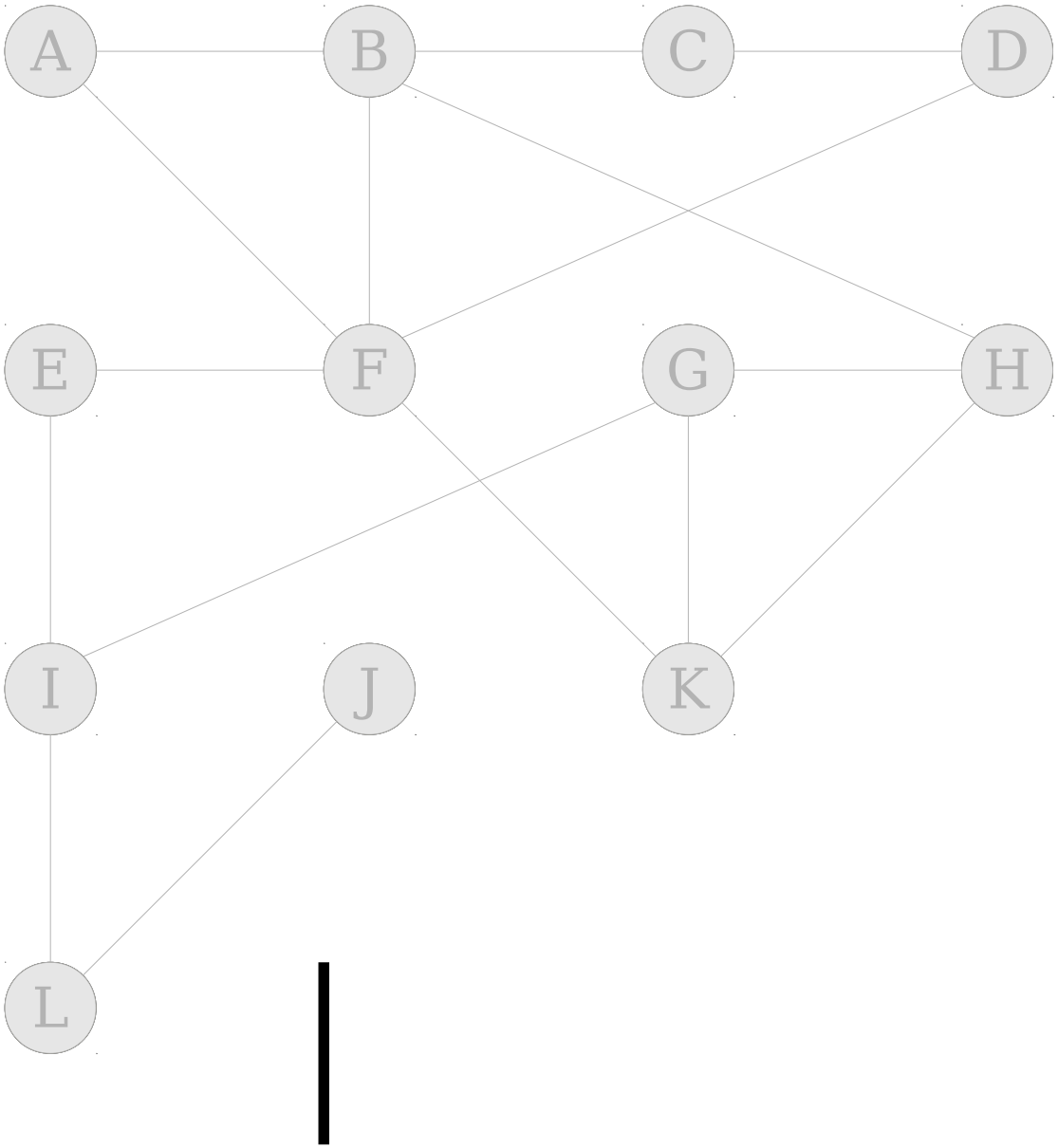
Breadth-First Search



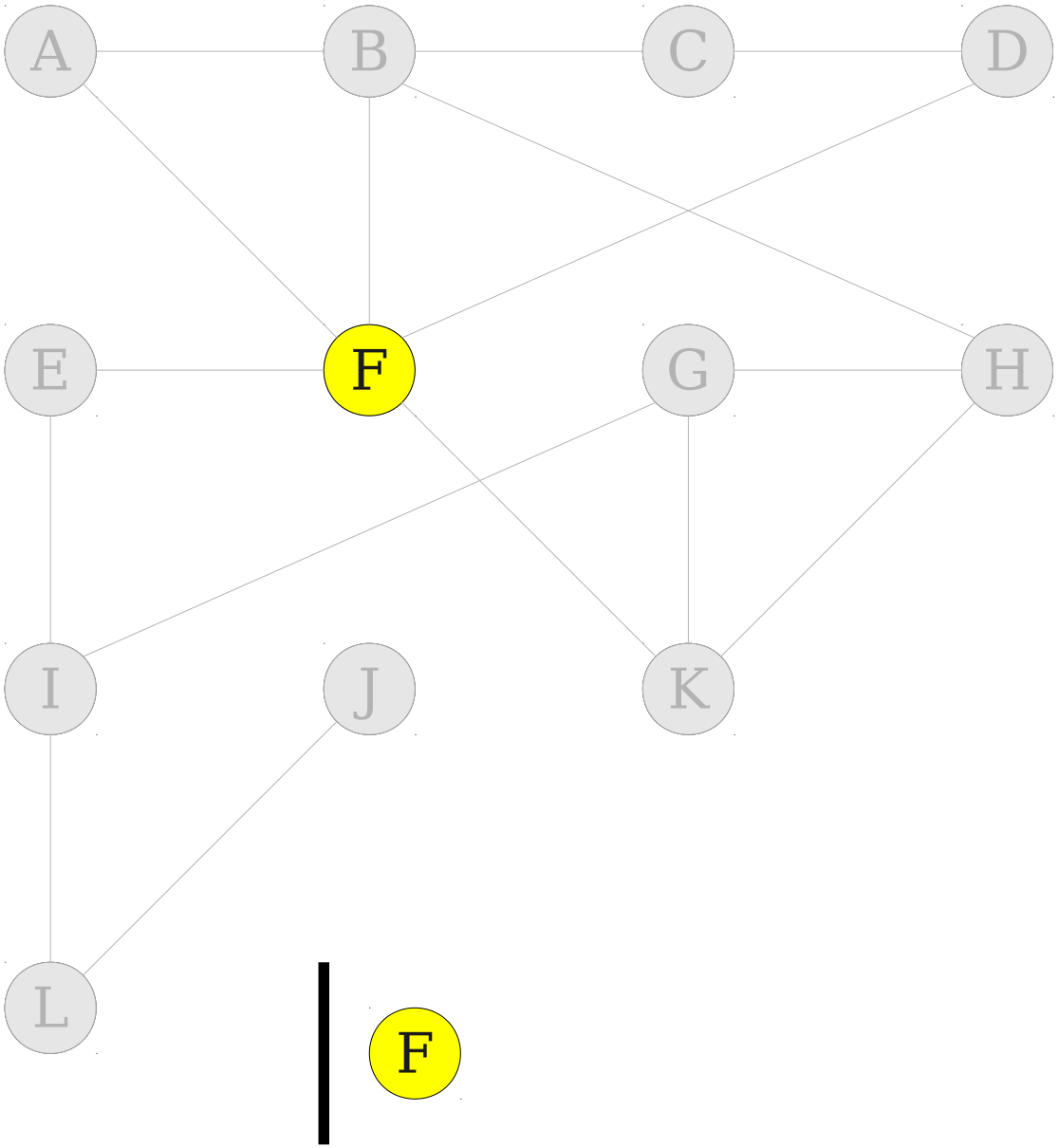
Breadth-First Search



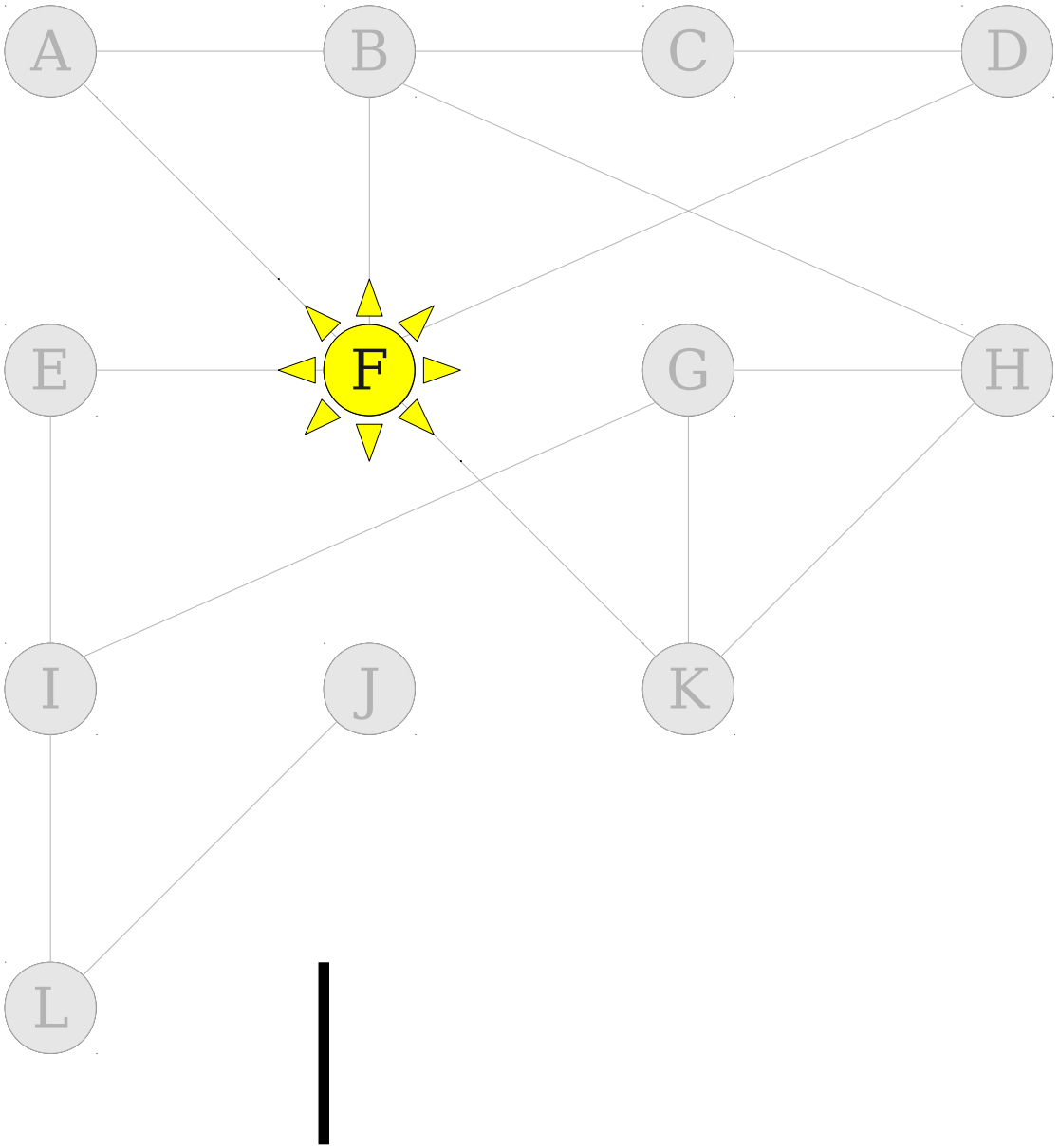
Breadth-First Search



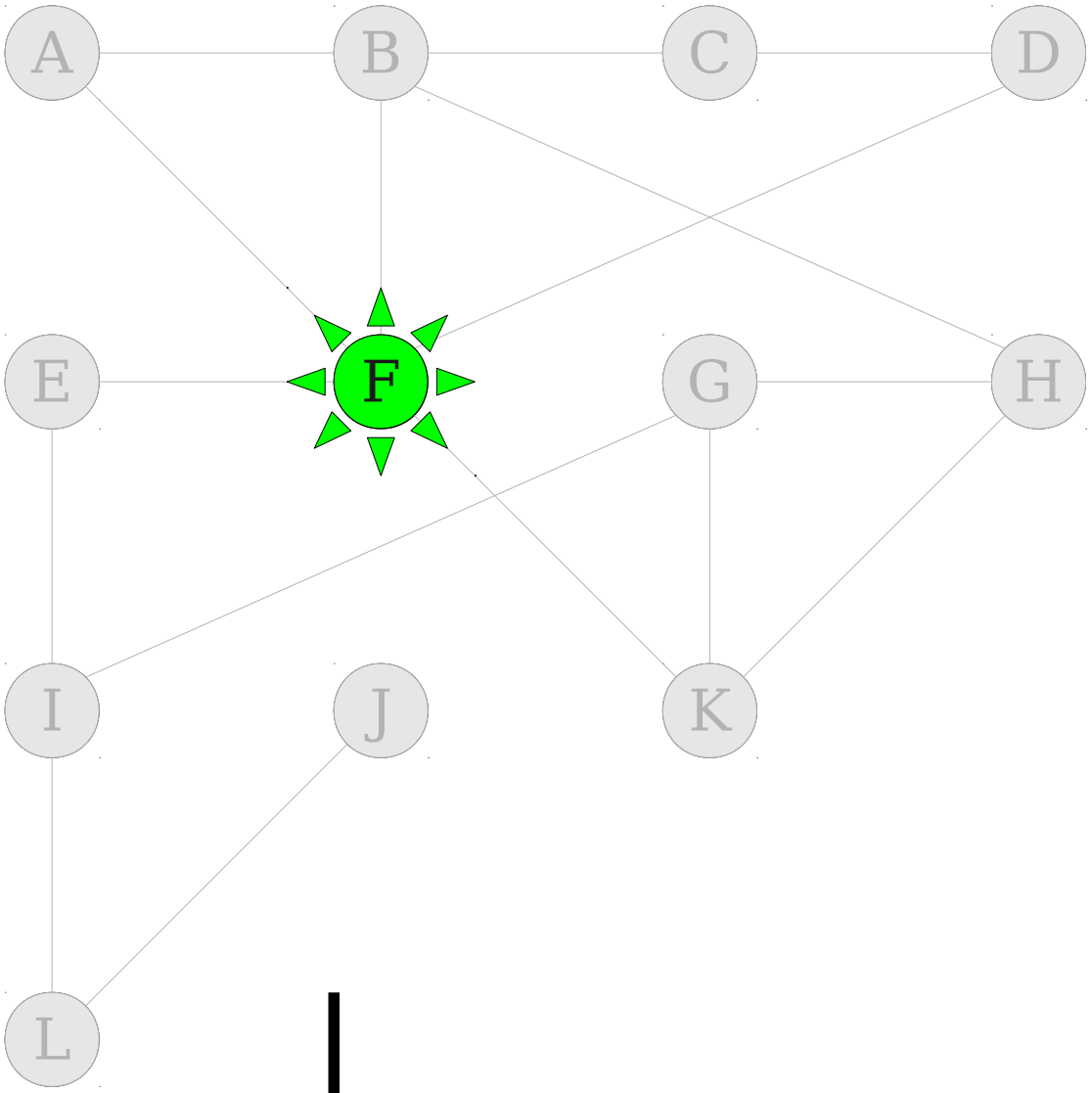
Breadth-First Search



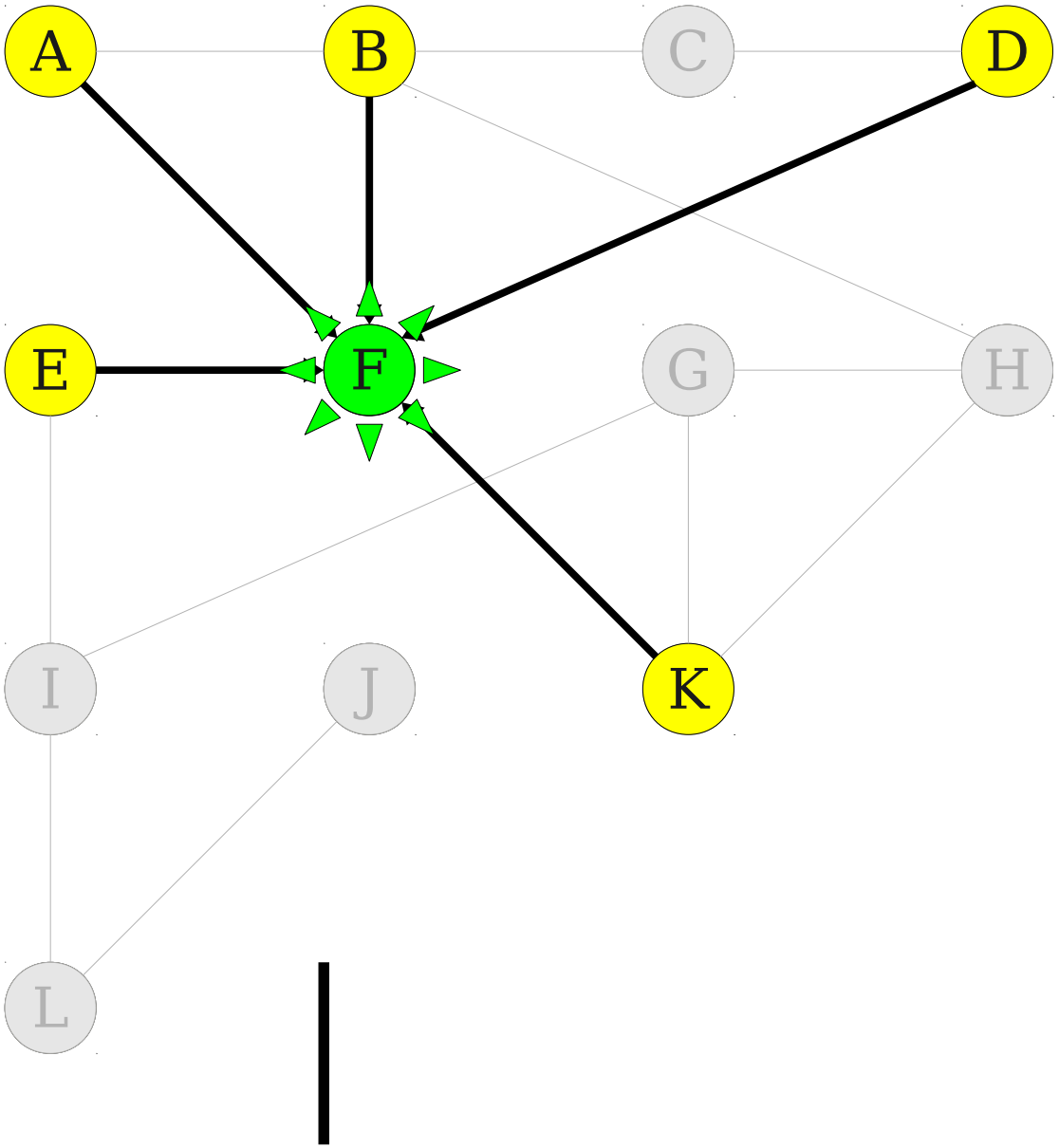
Breadth-First Search



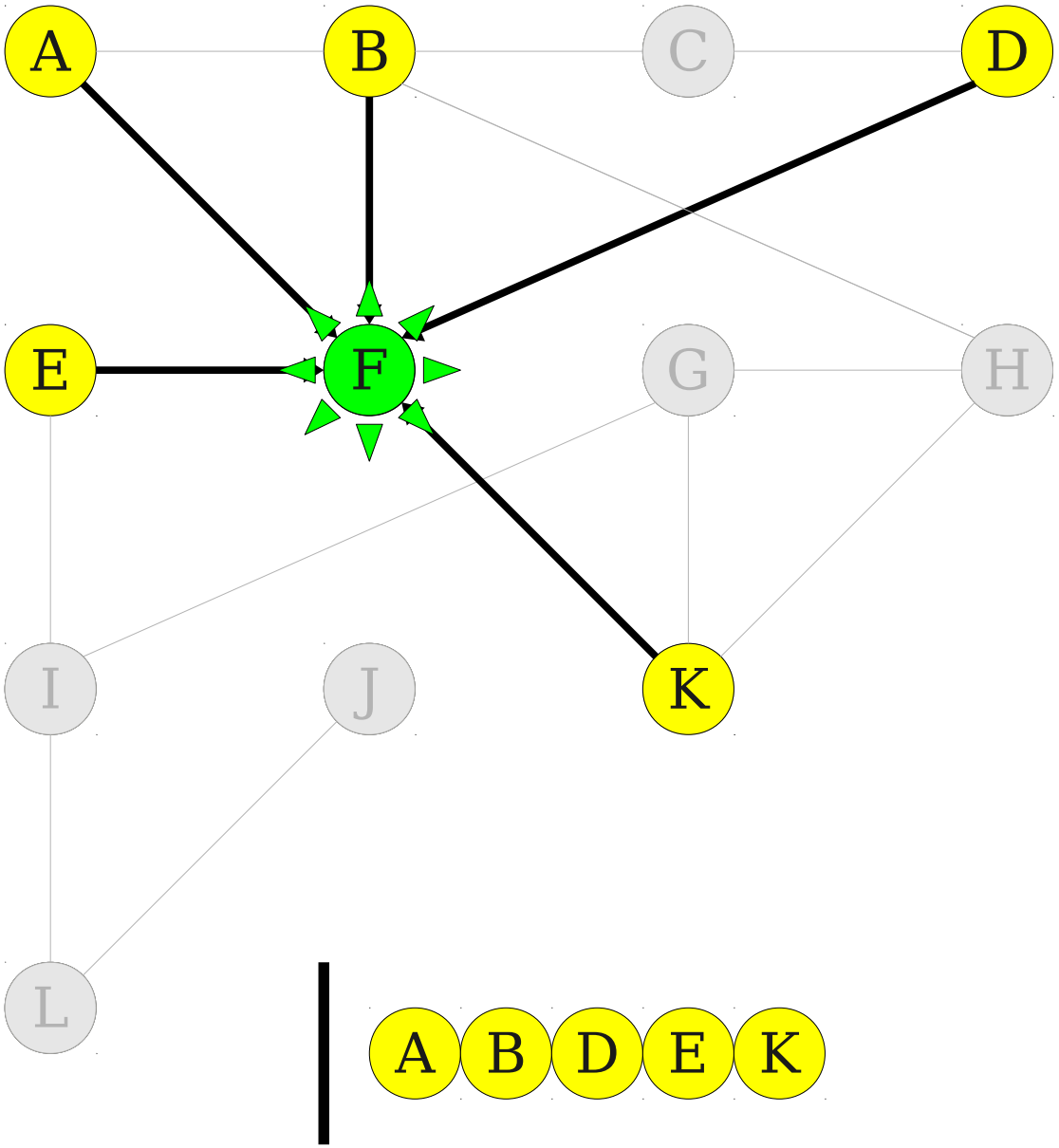
Breadth-First Search



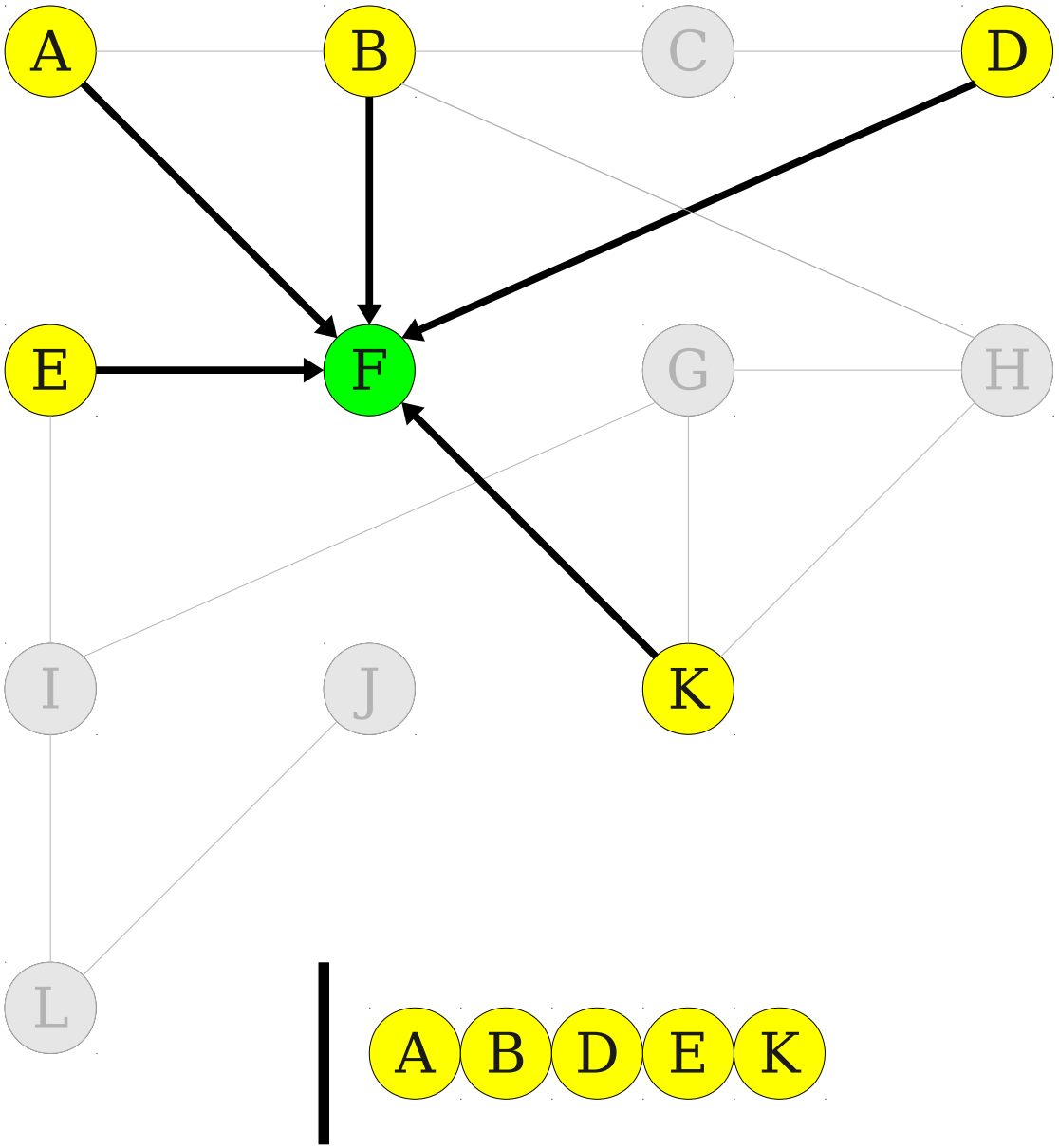
Breadth-First Search



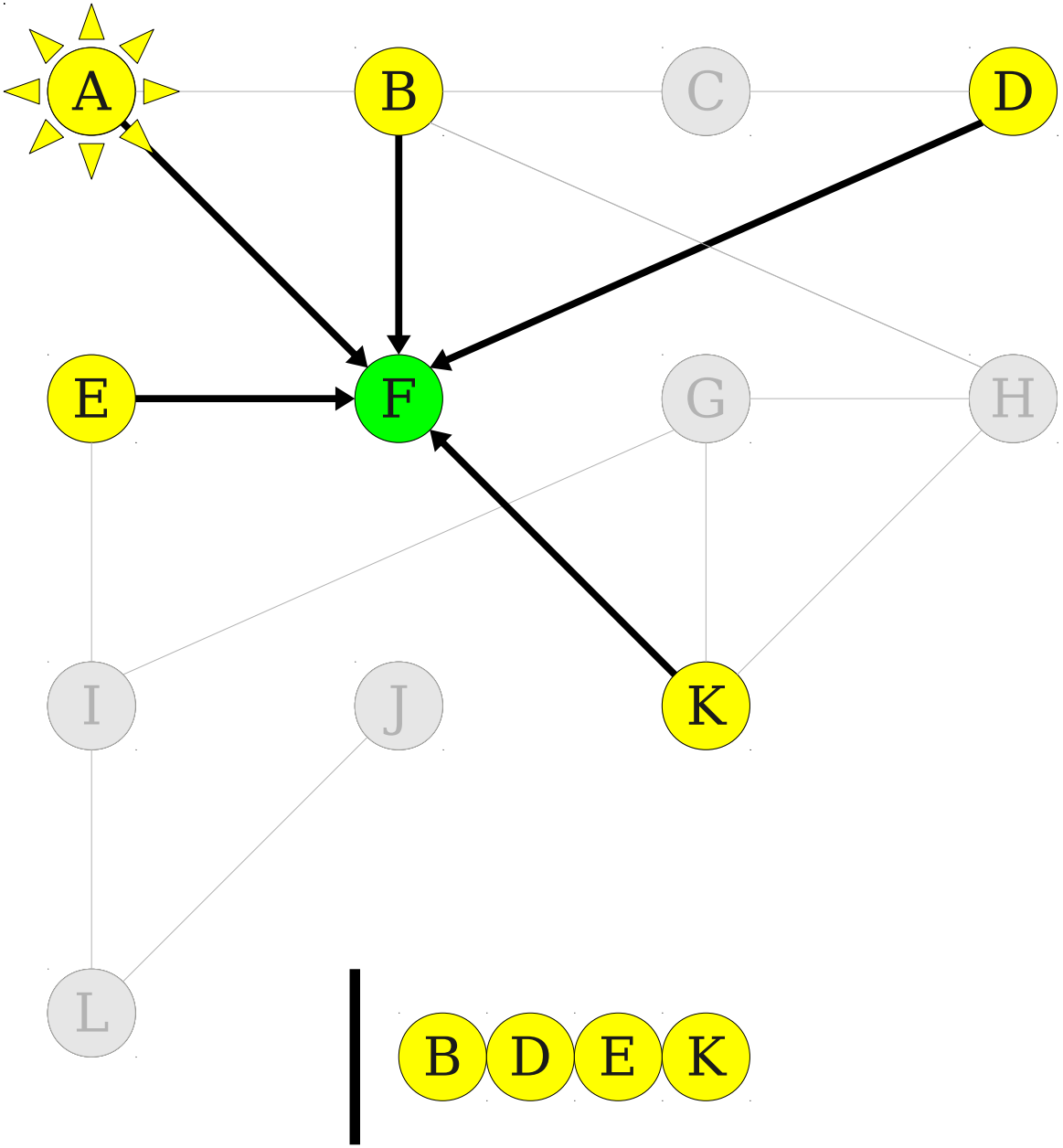
Breadth-First Search



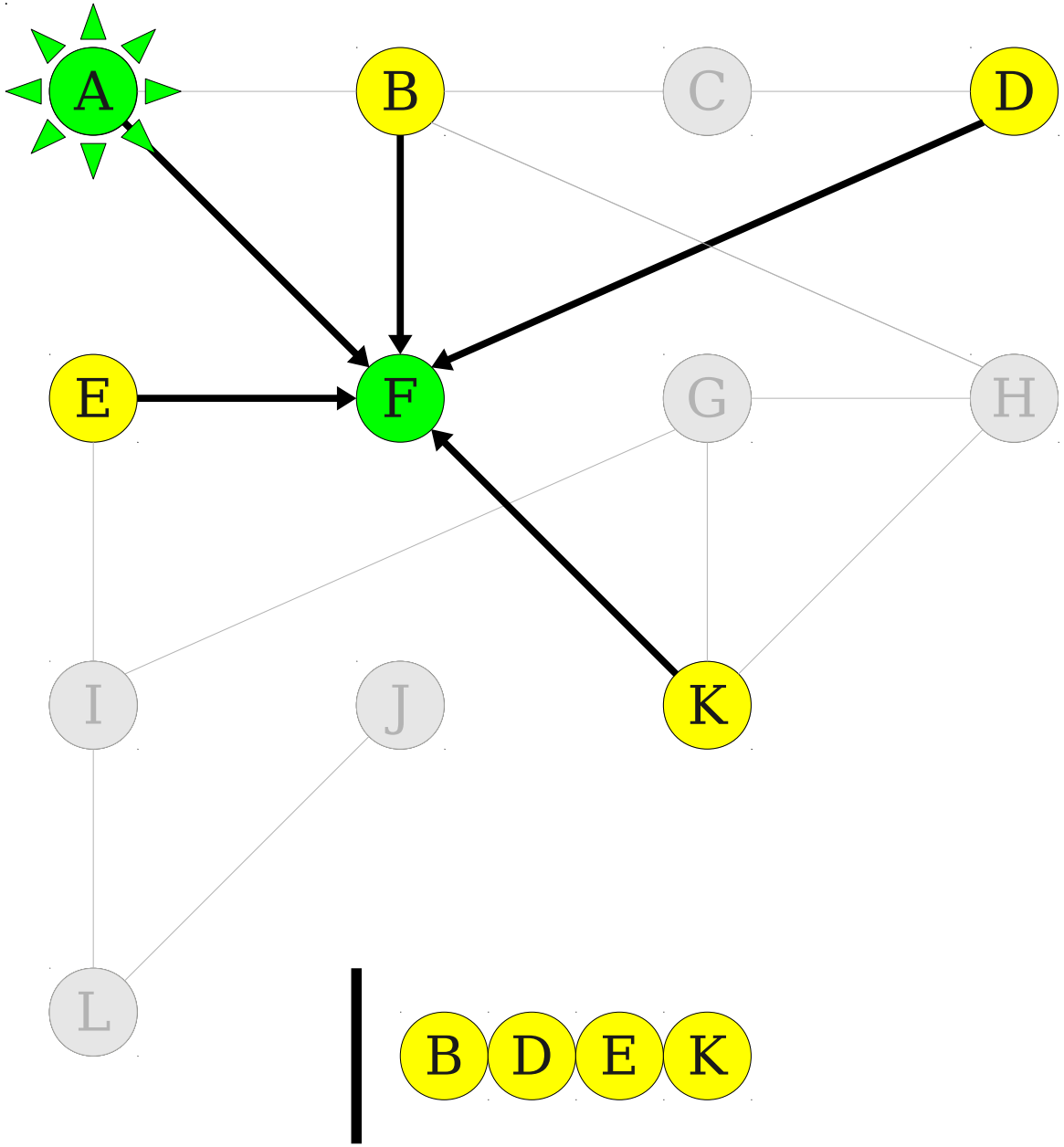
Breadth-First Search



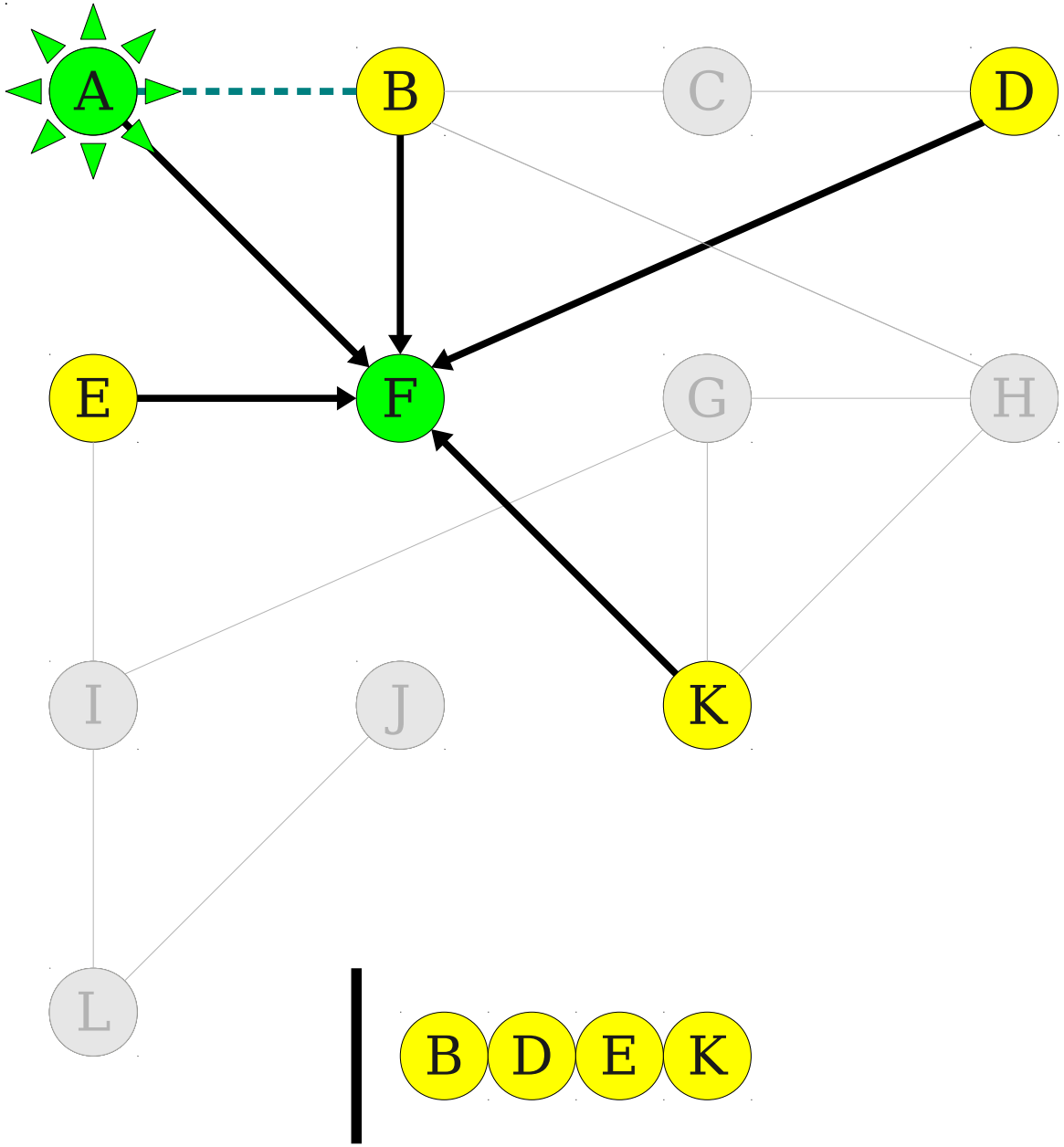
Breadth-First Search



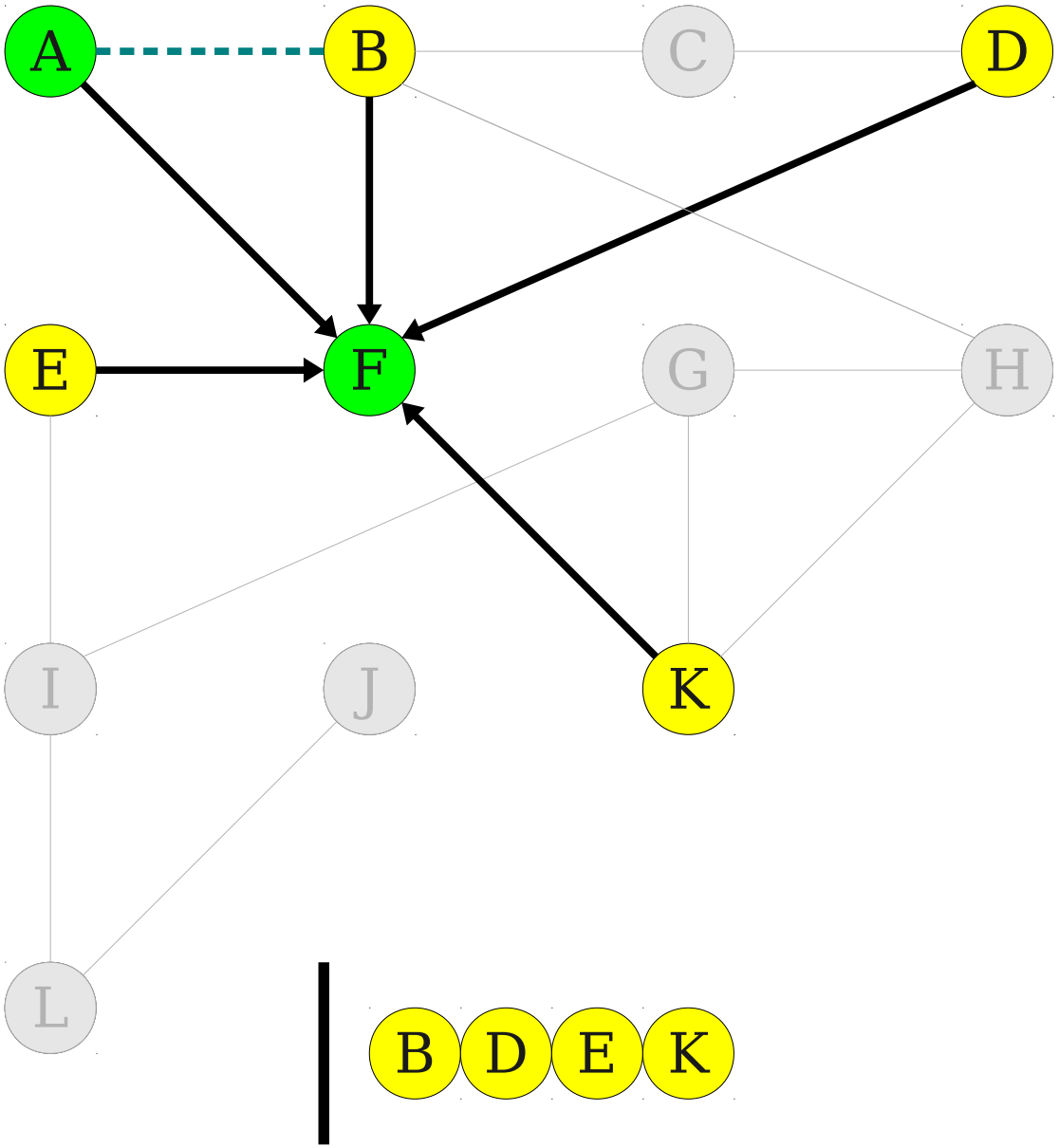
Breadth-First Search



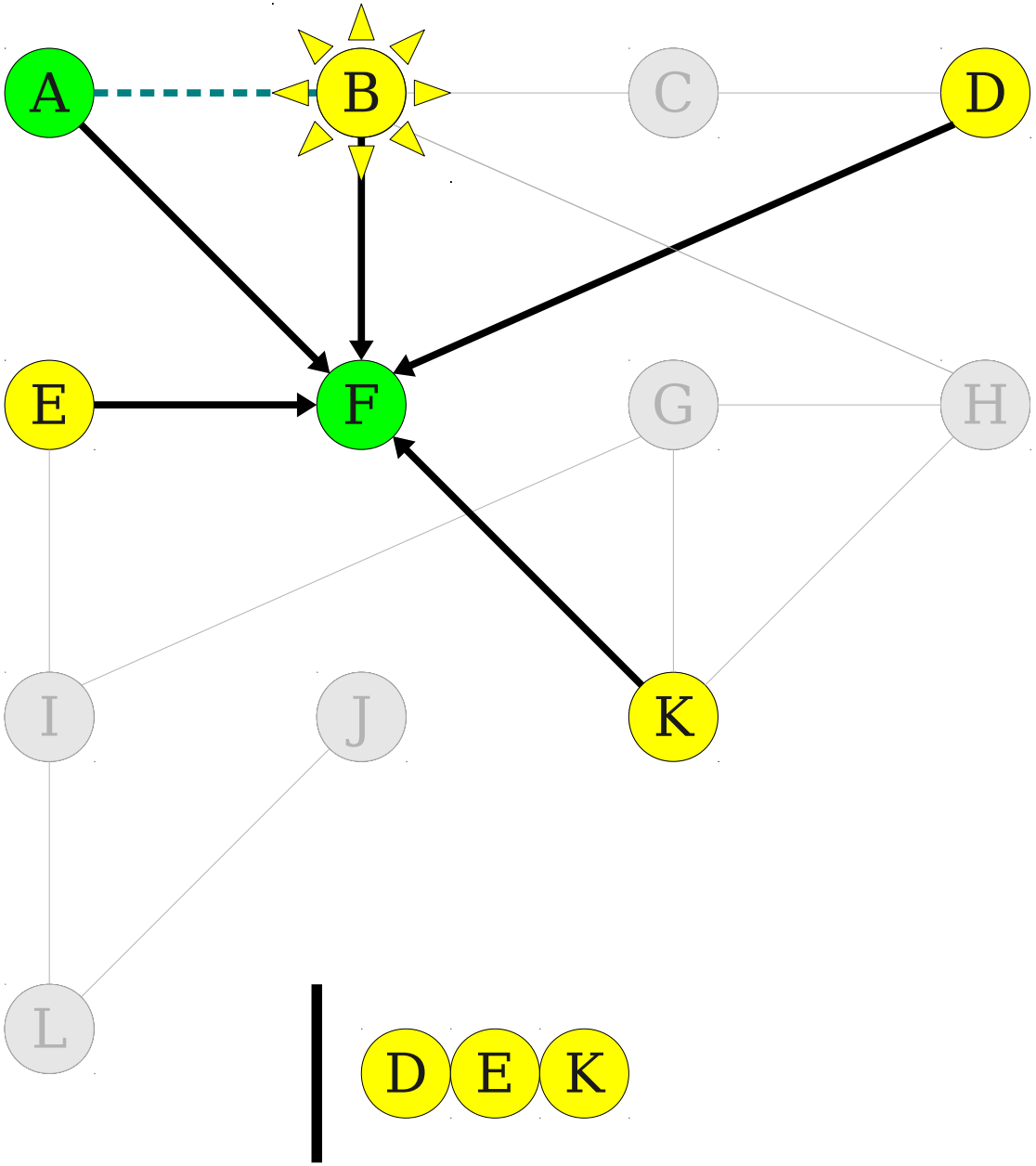
Breadth-First Search



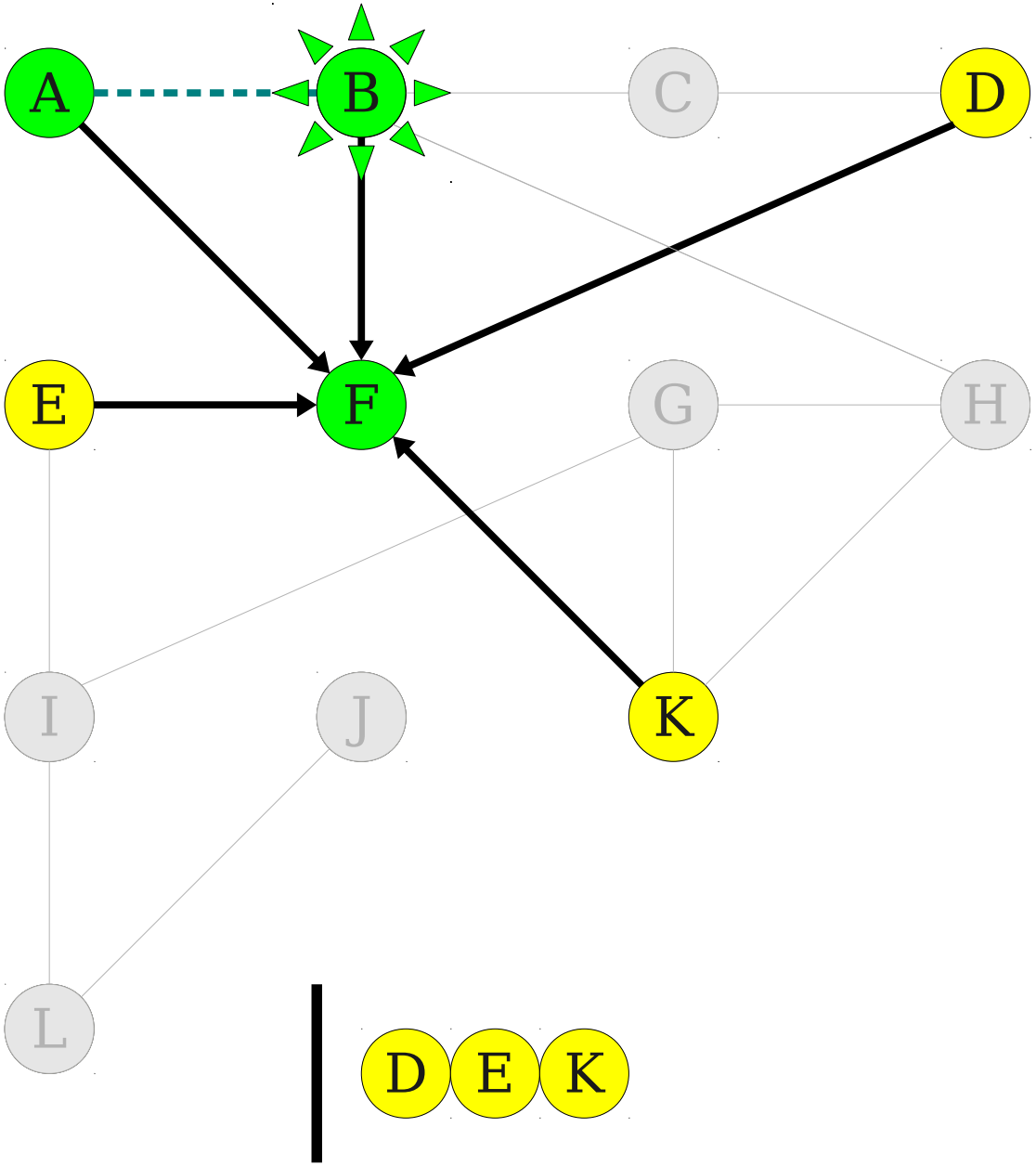
Breadth-First Search



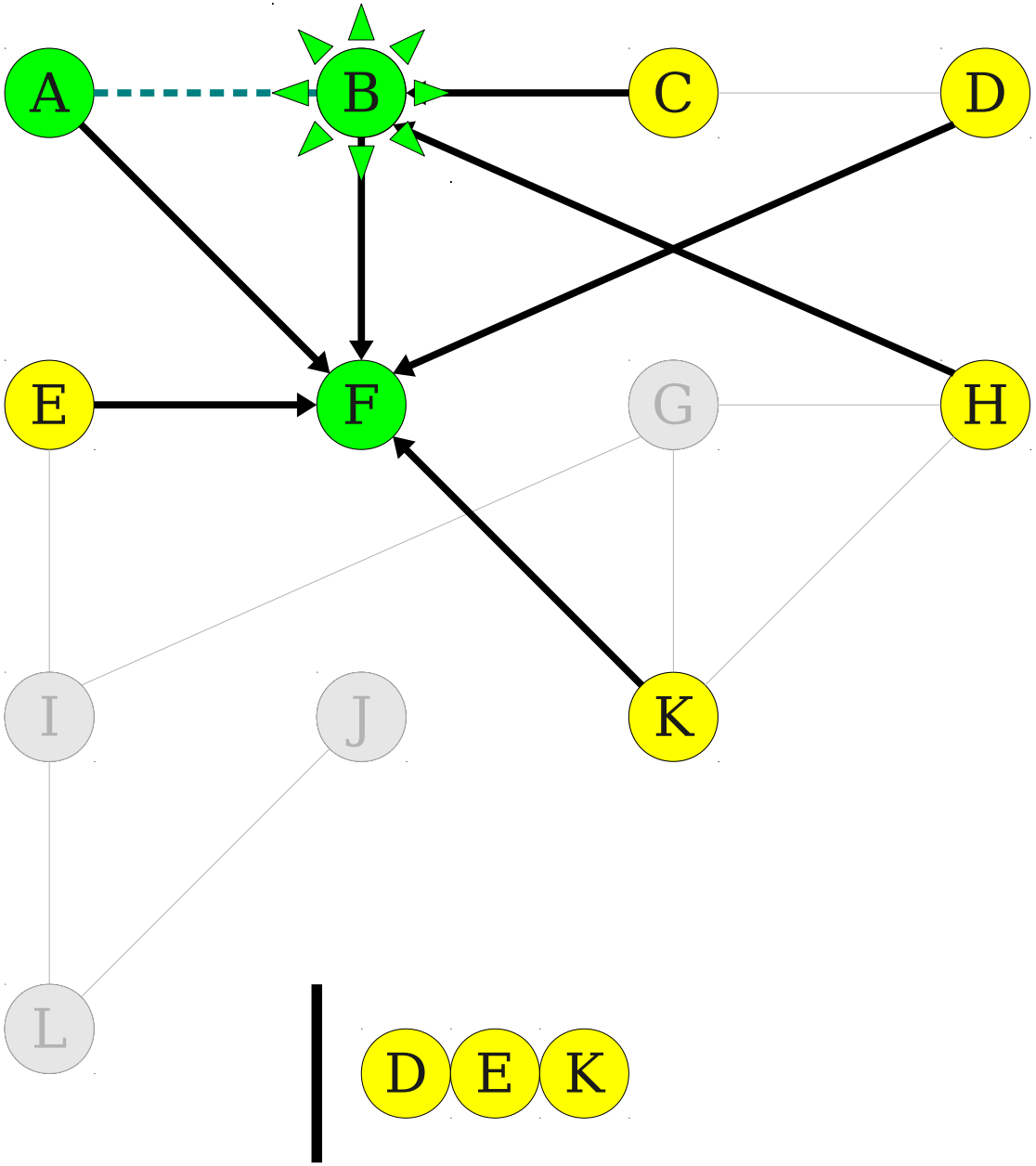
Breadth-First Search



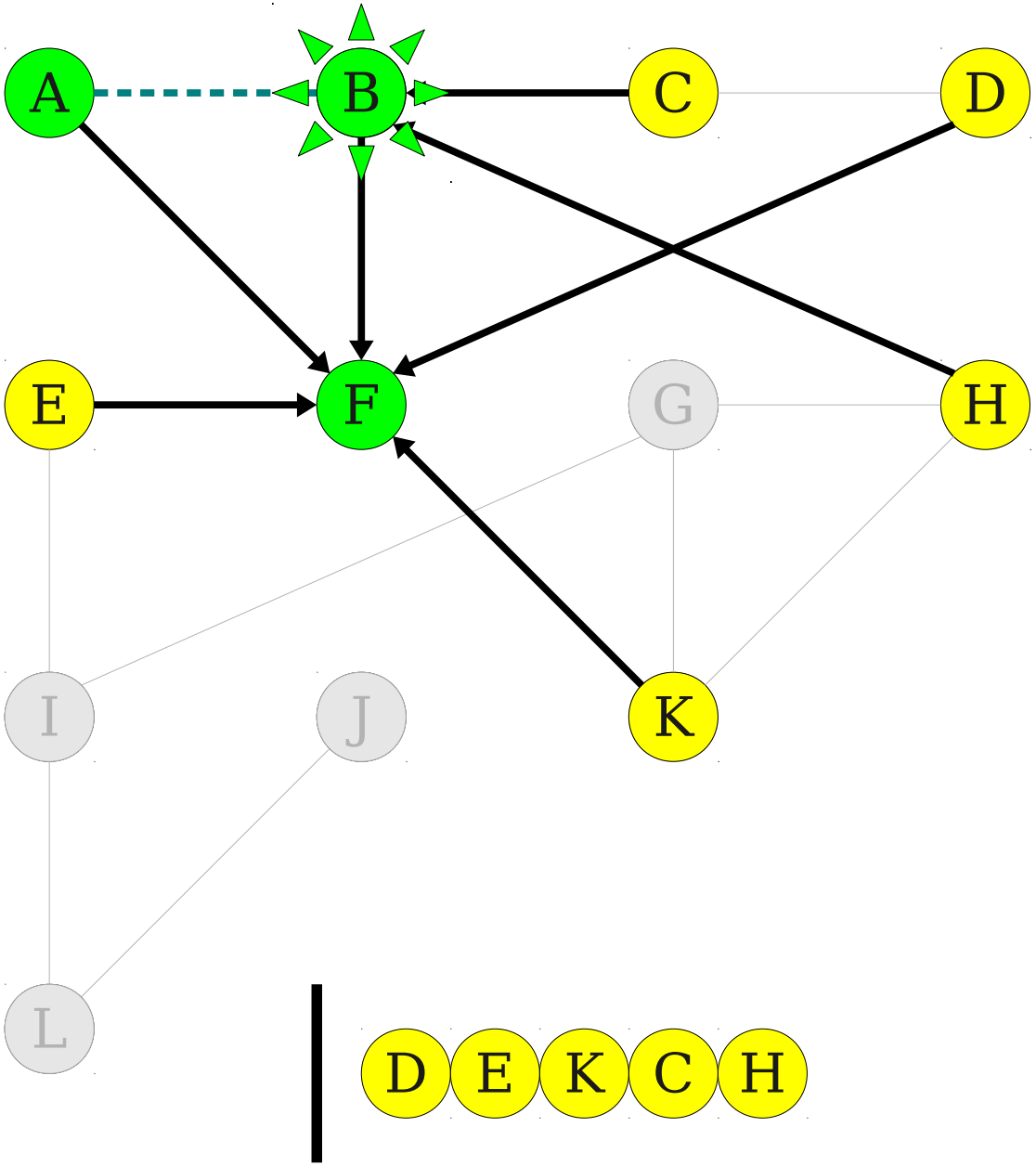
Breadth-First Search



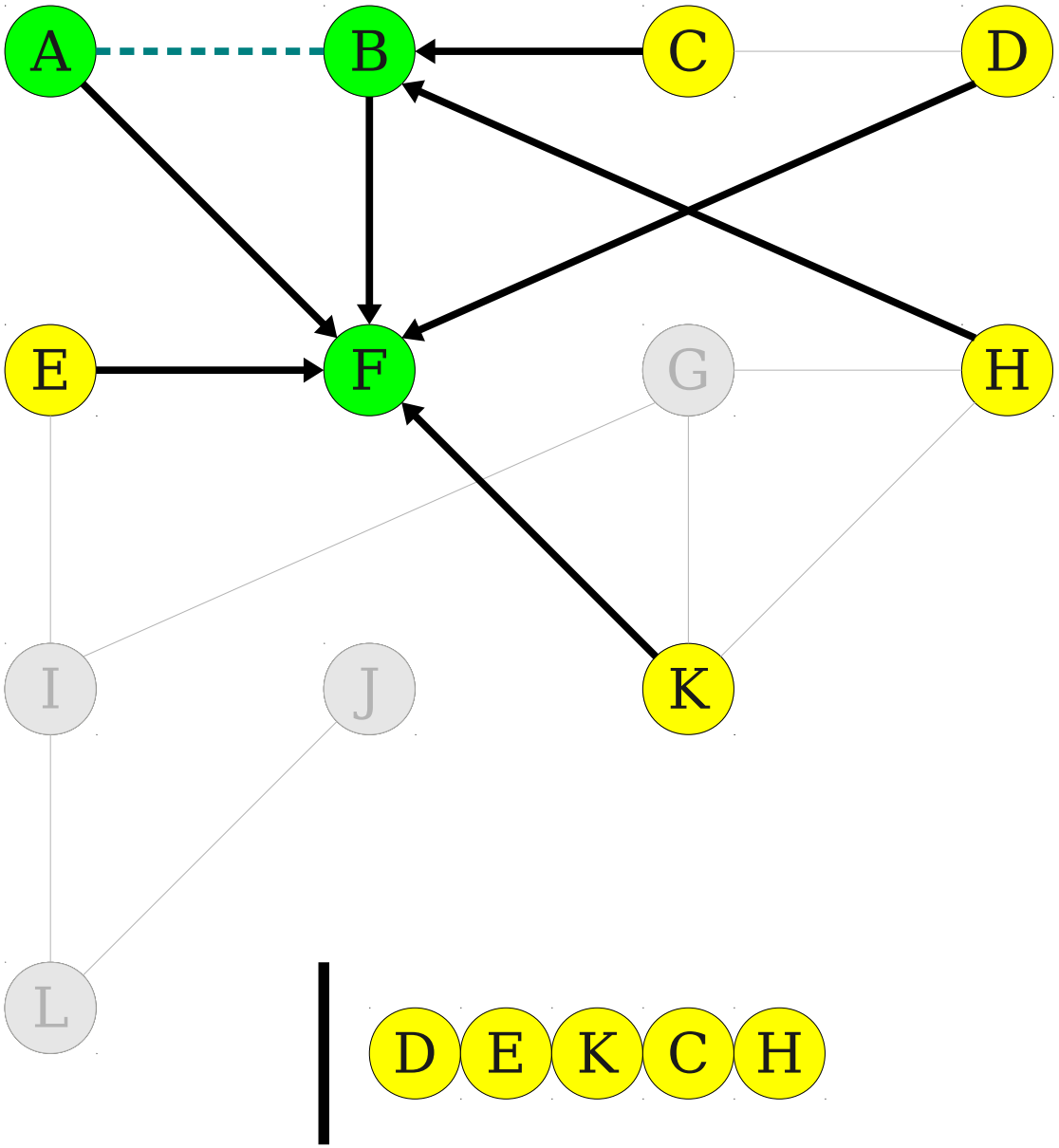
Breadth-First Search



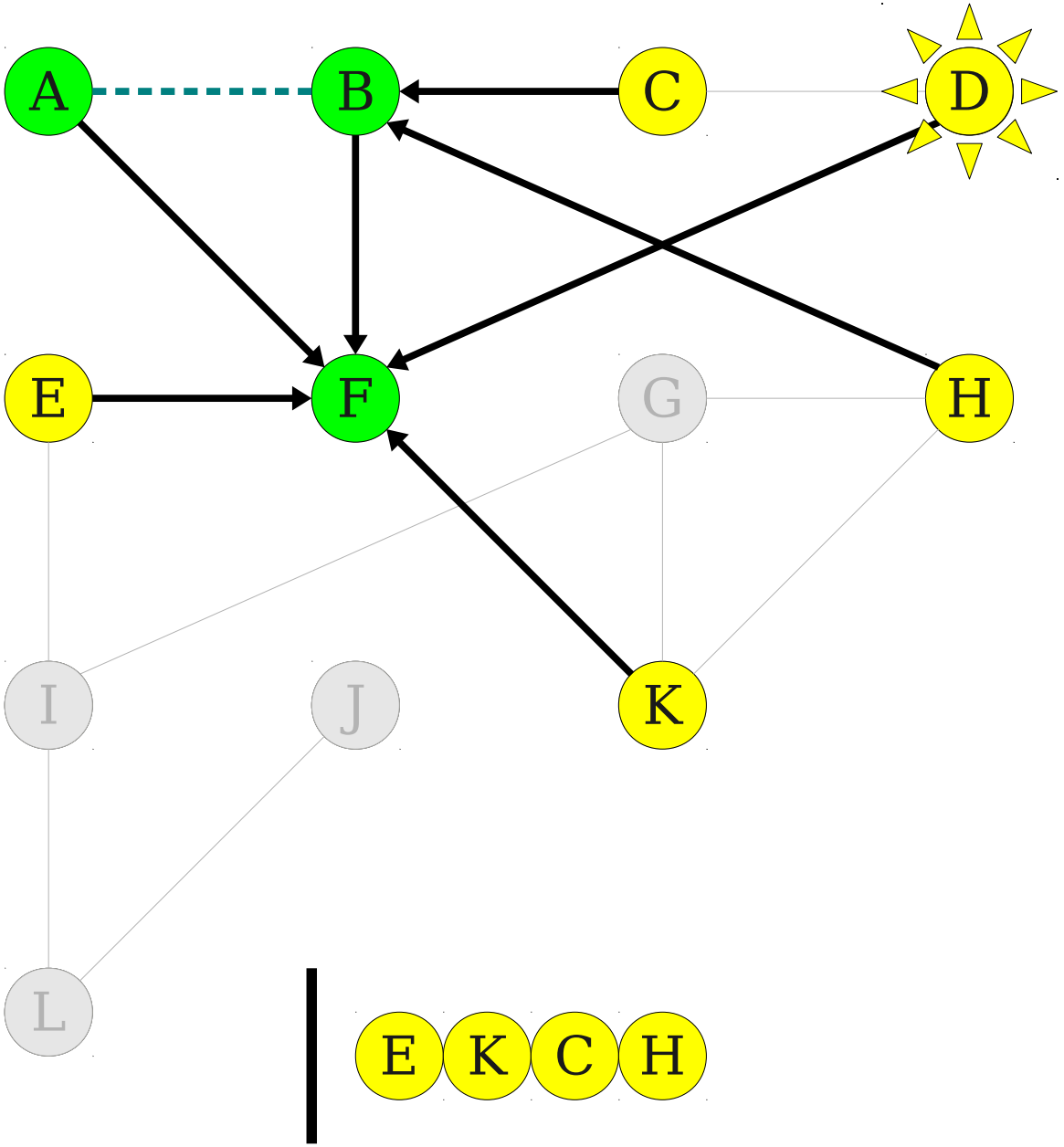
Breadth-First Search



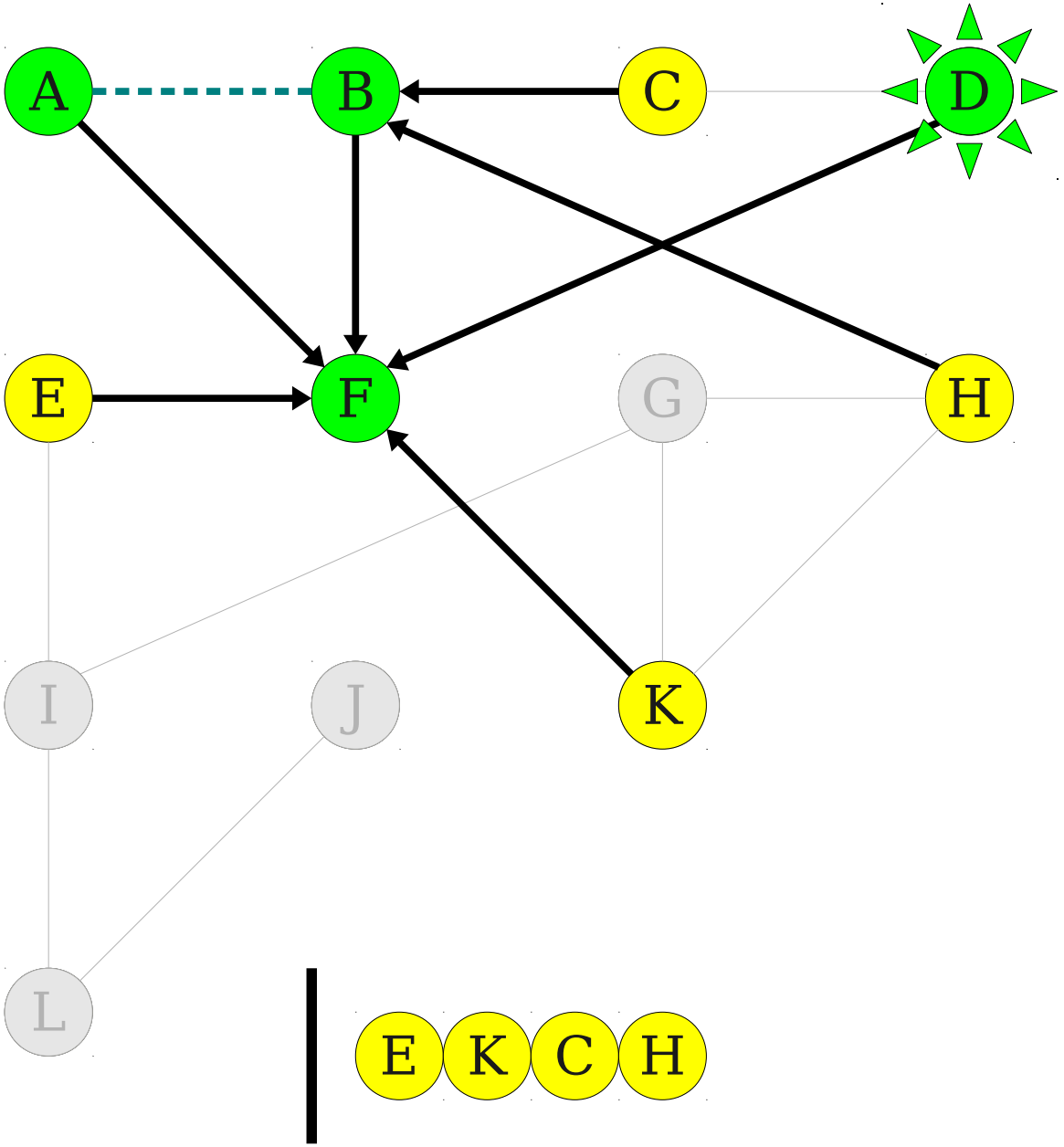
Breadth-First Search



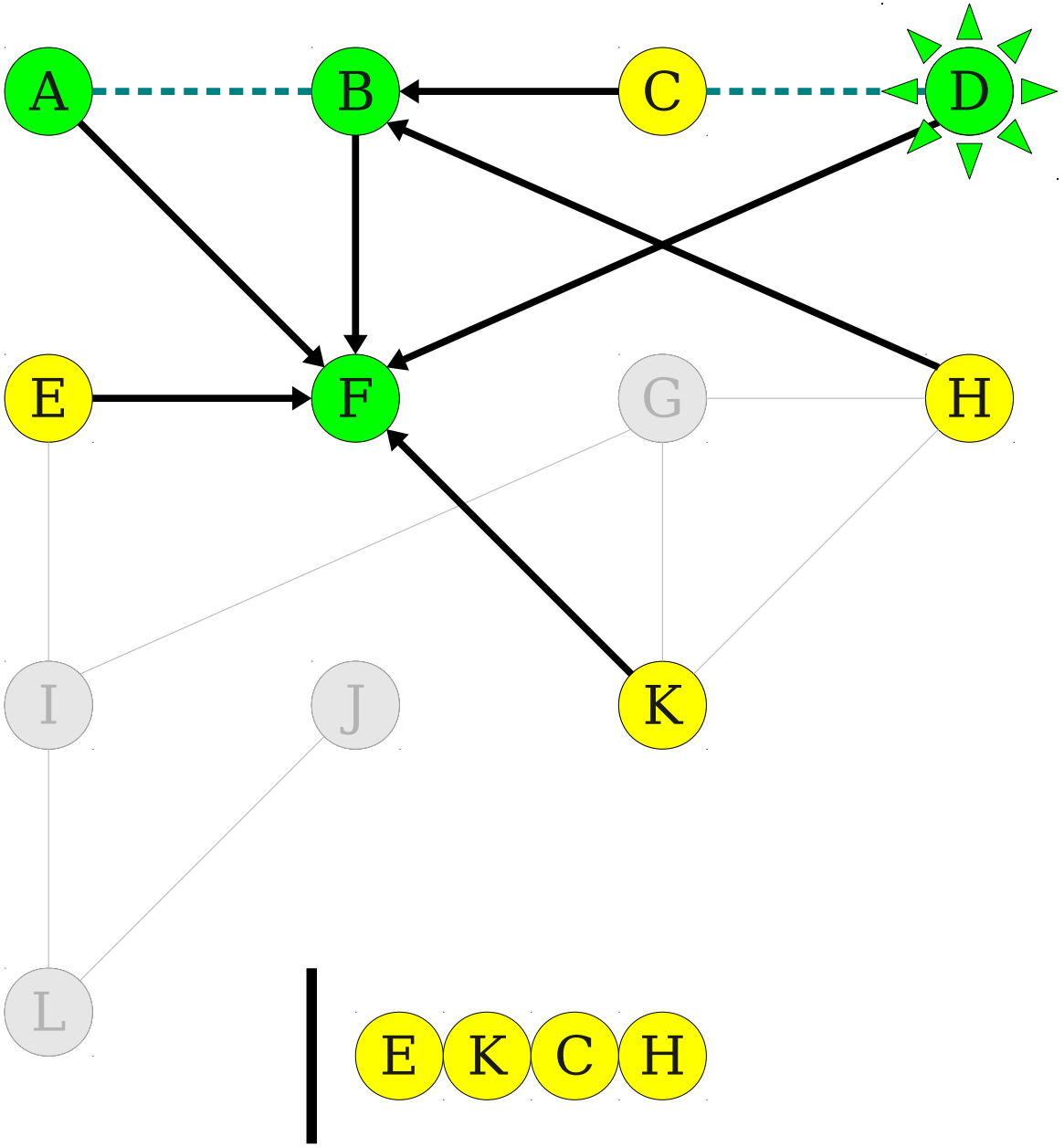
Breadth-First Search



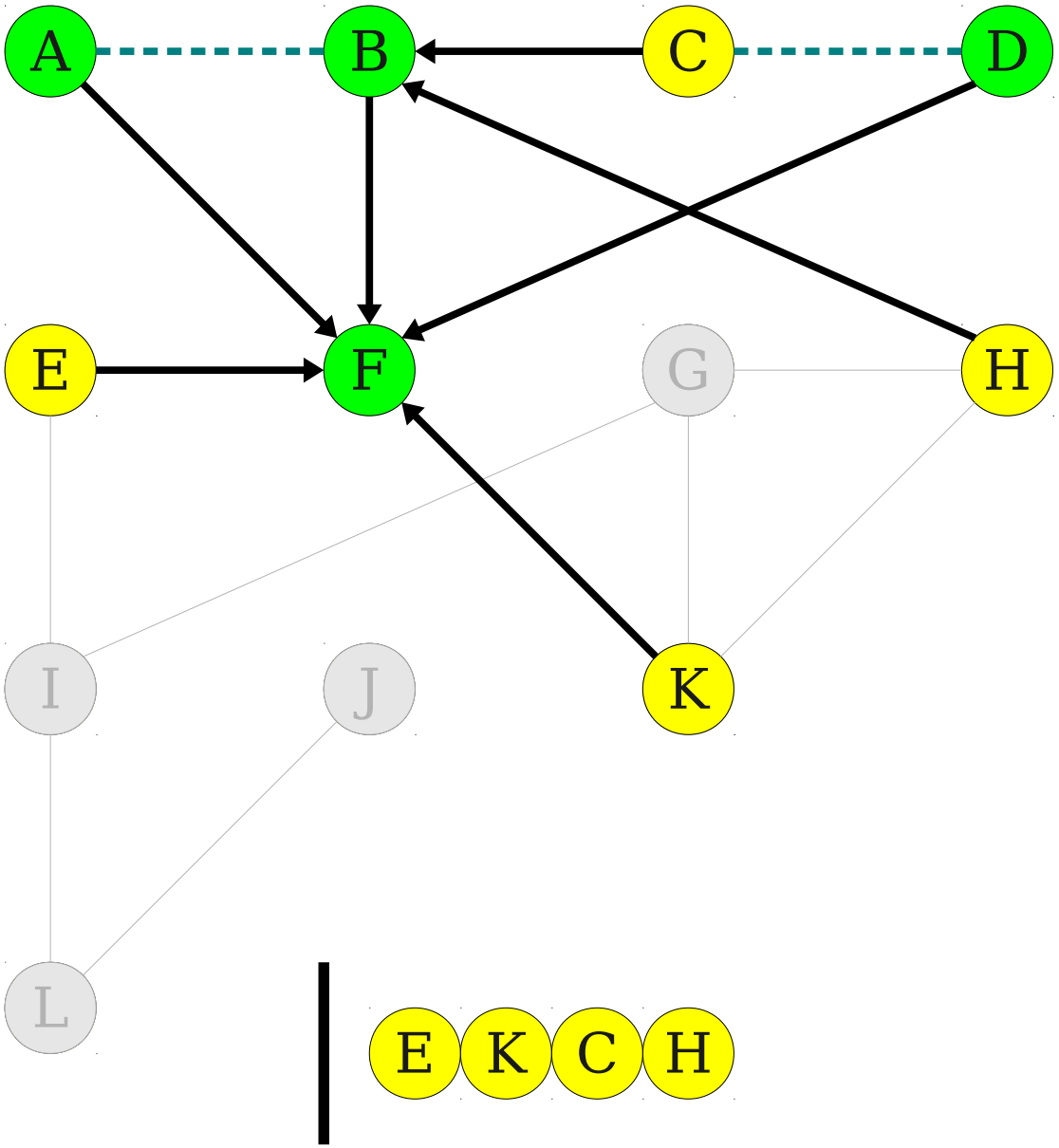
Breadth-First Search



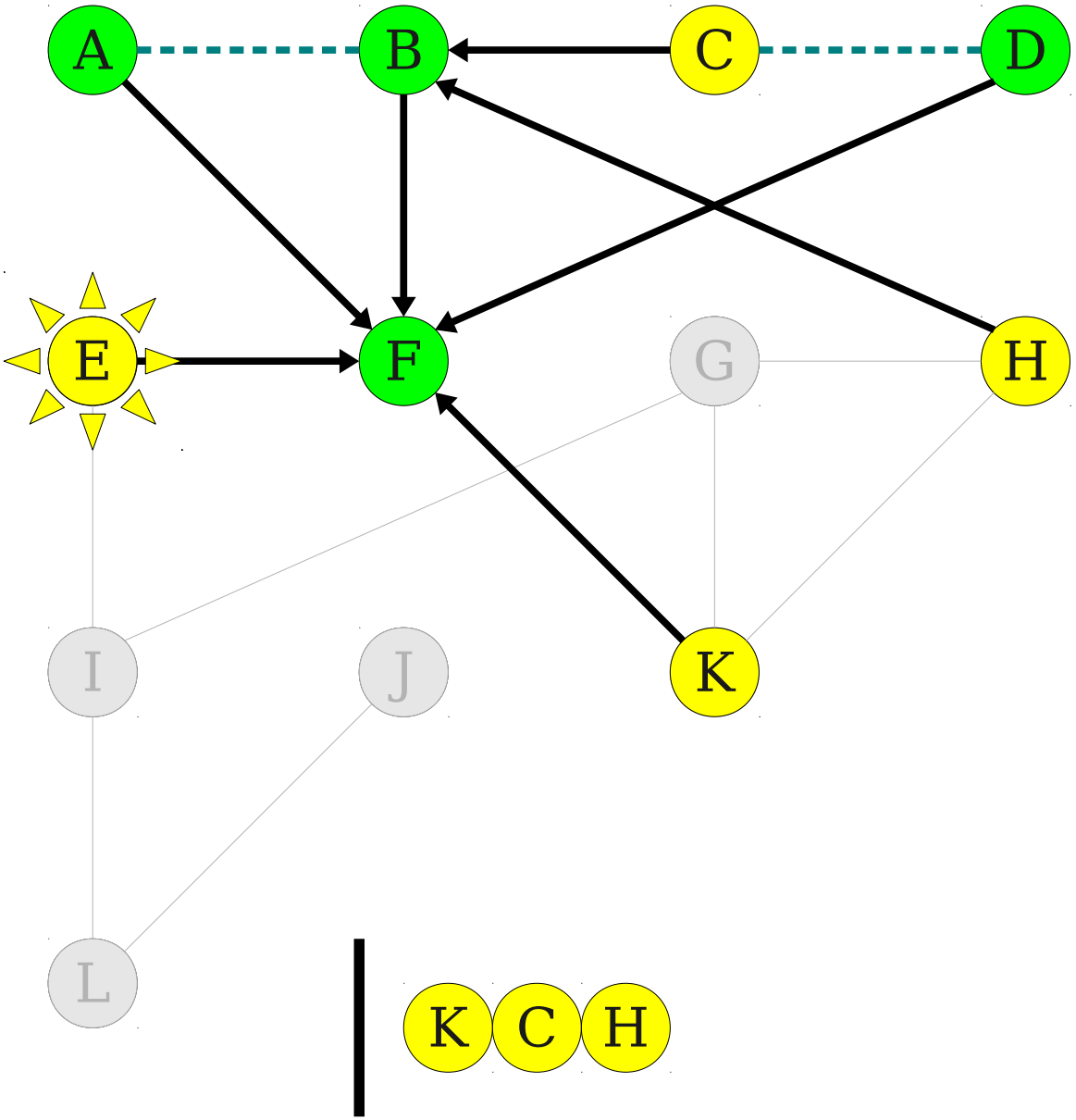
Breadth-First Search



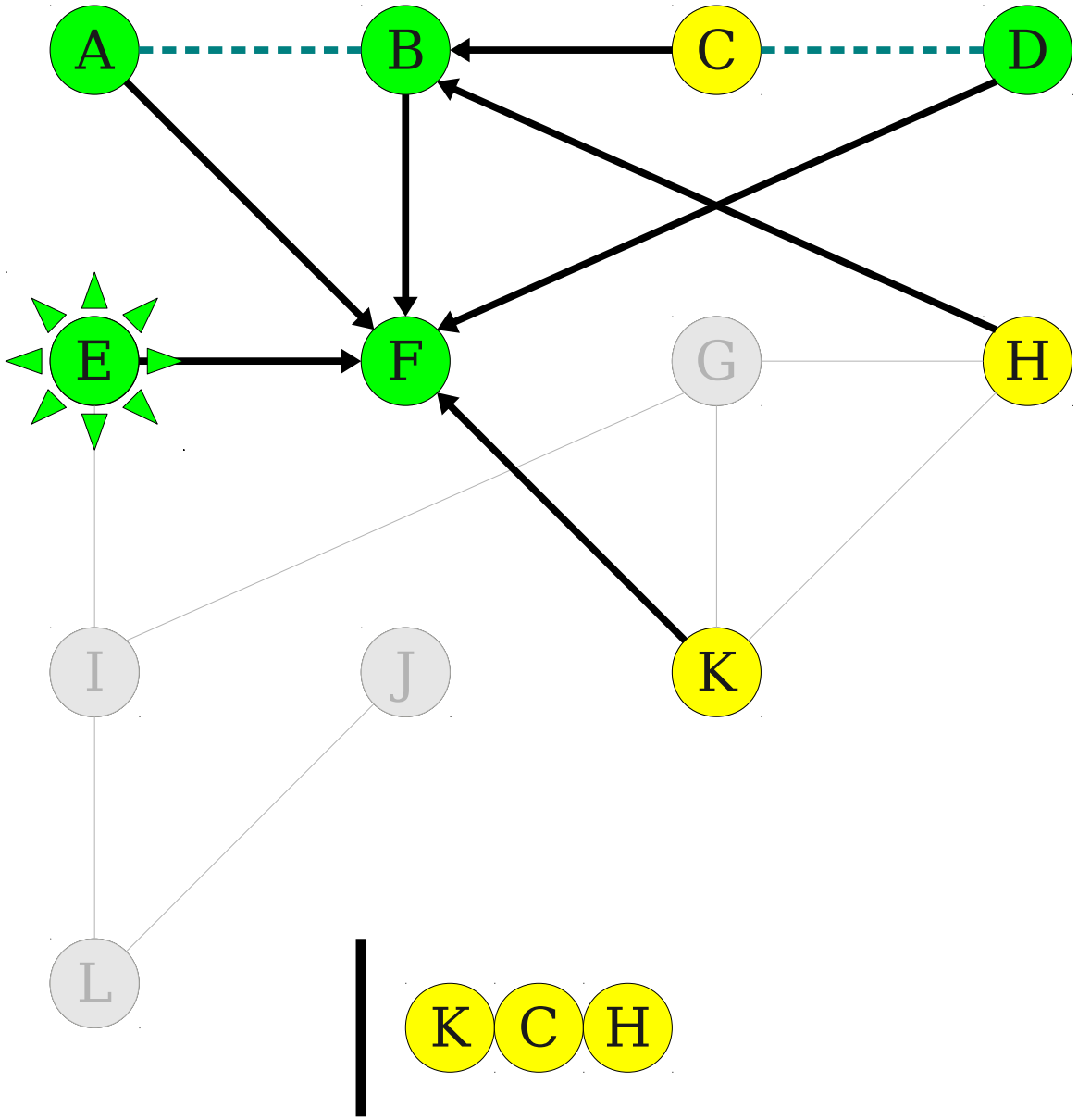
Breadth-First Search



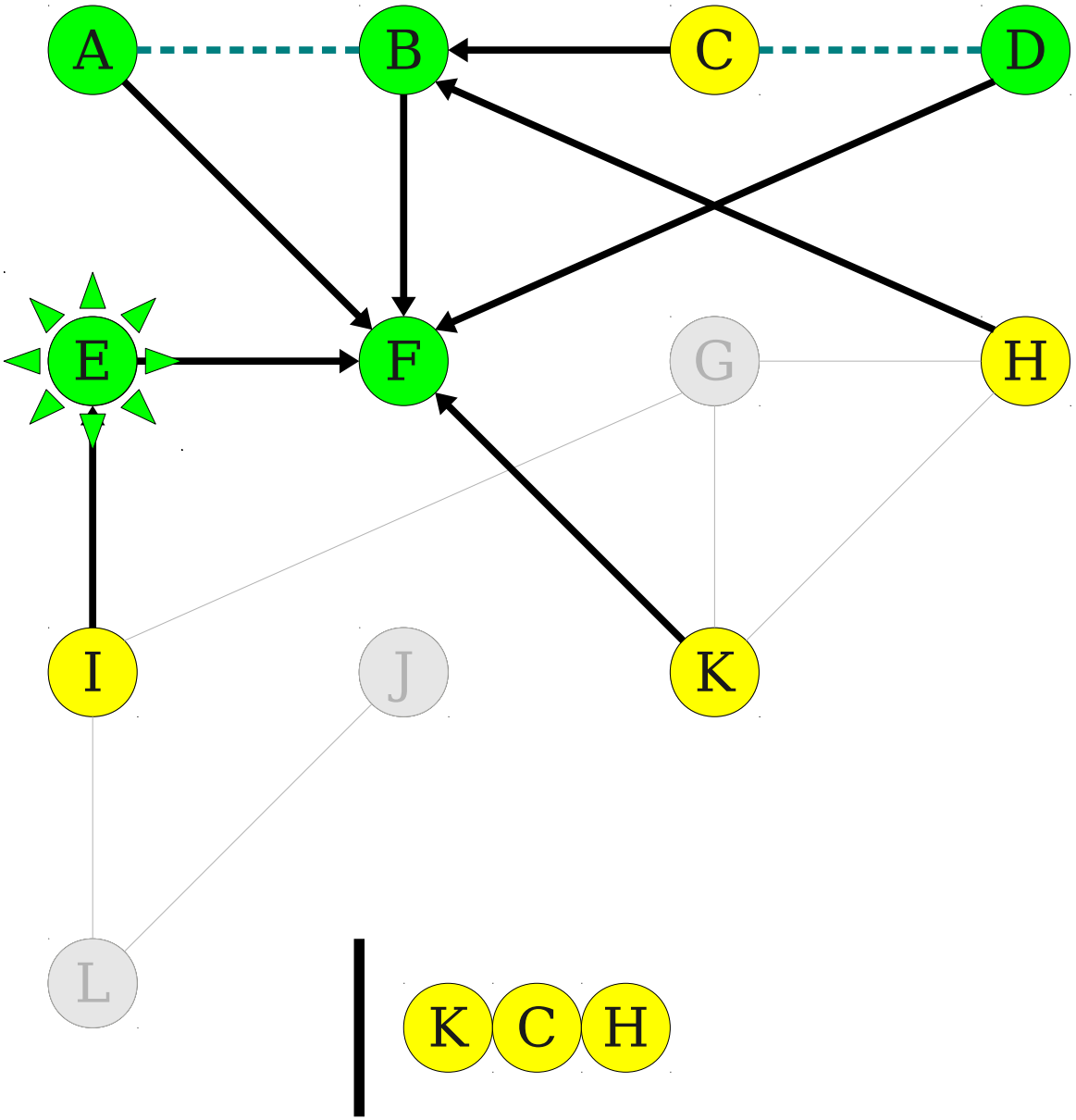
Breadth-First Search



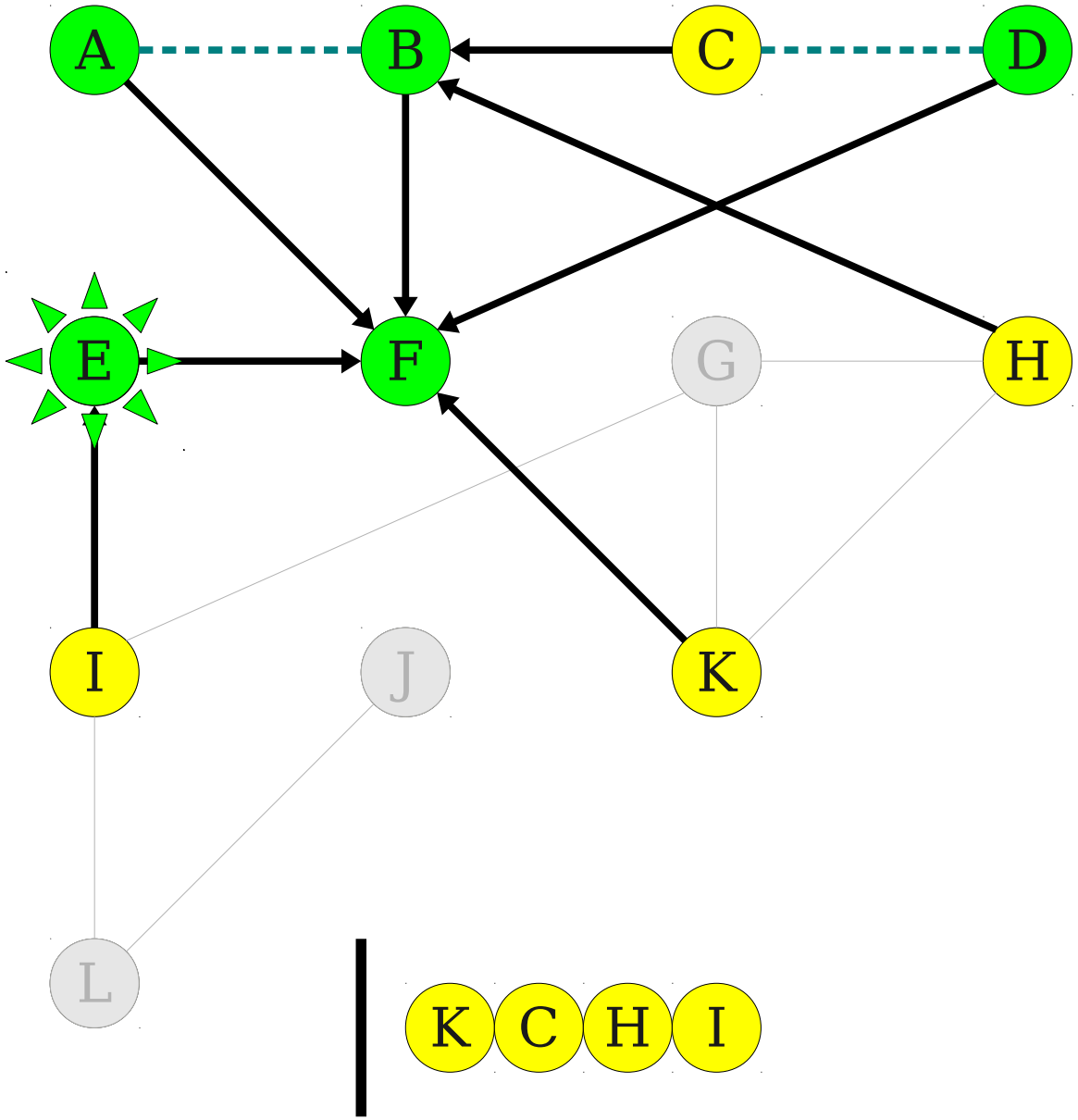
Breadth-First Search



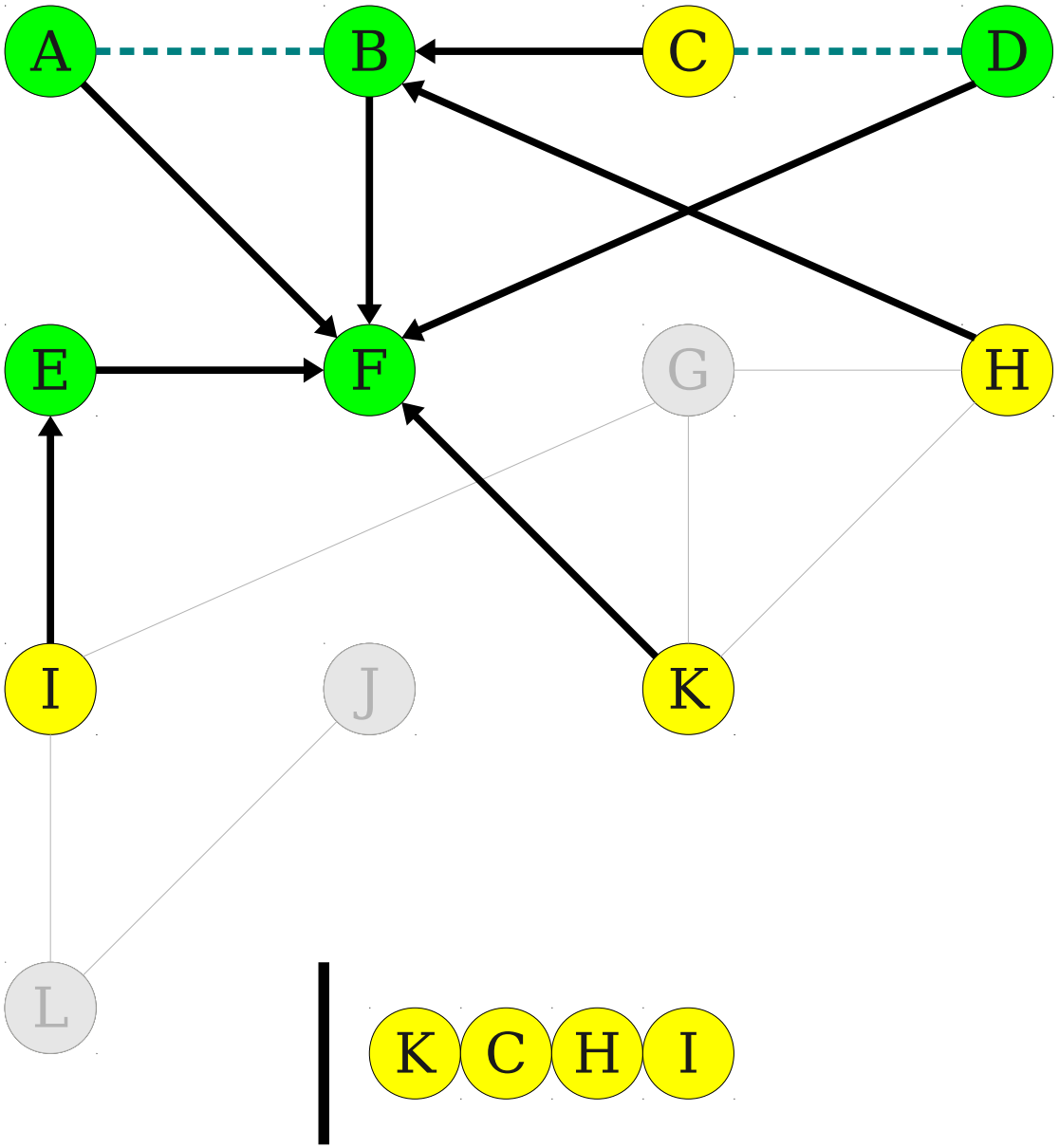
Breadth-First Search



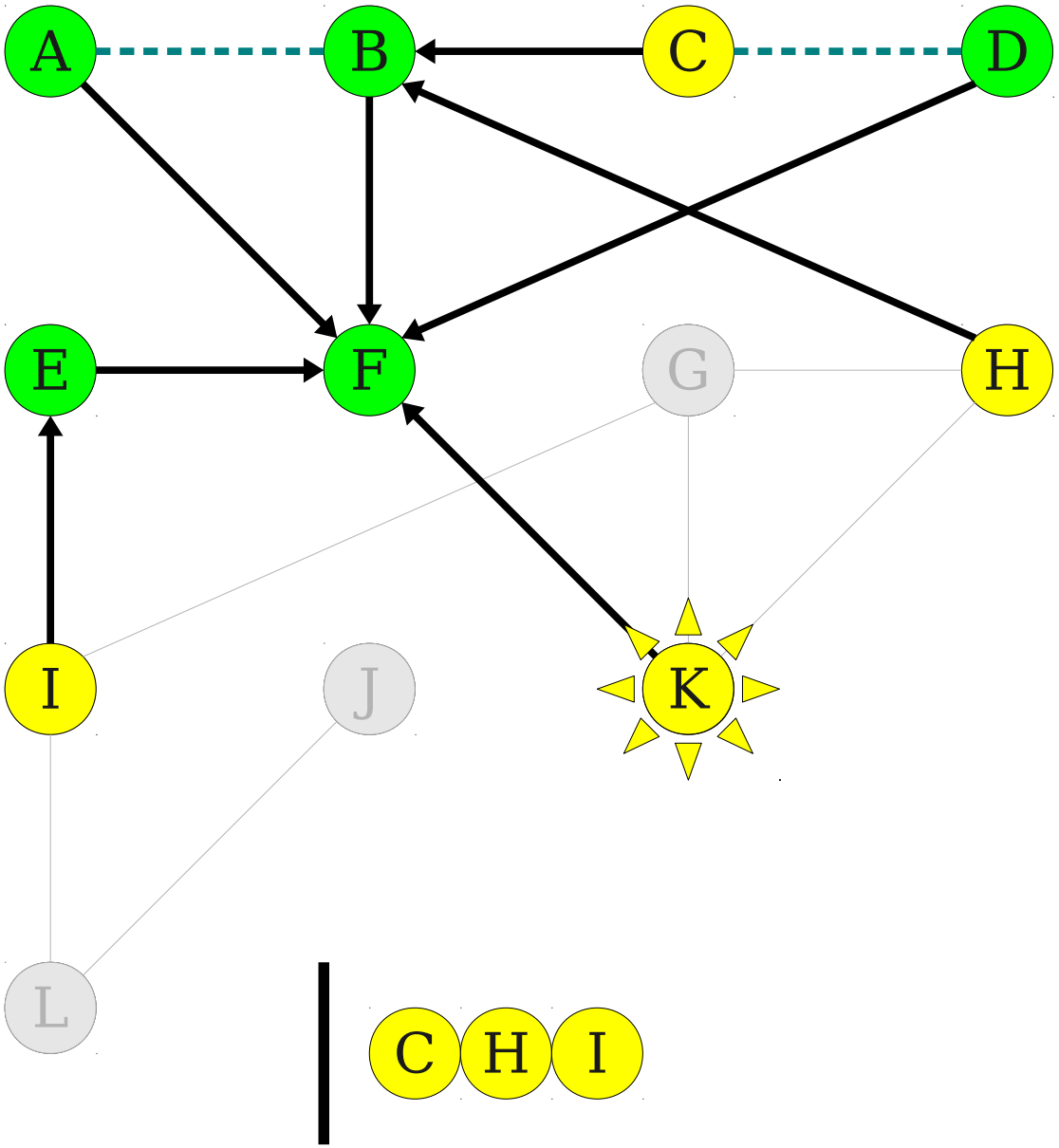
Breadth-First Search



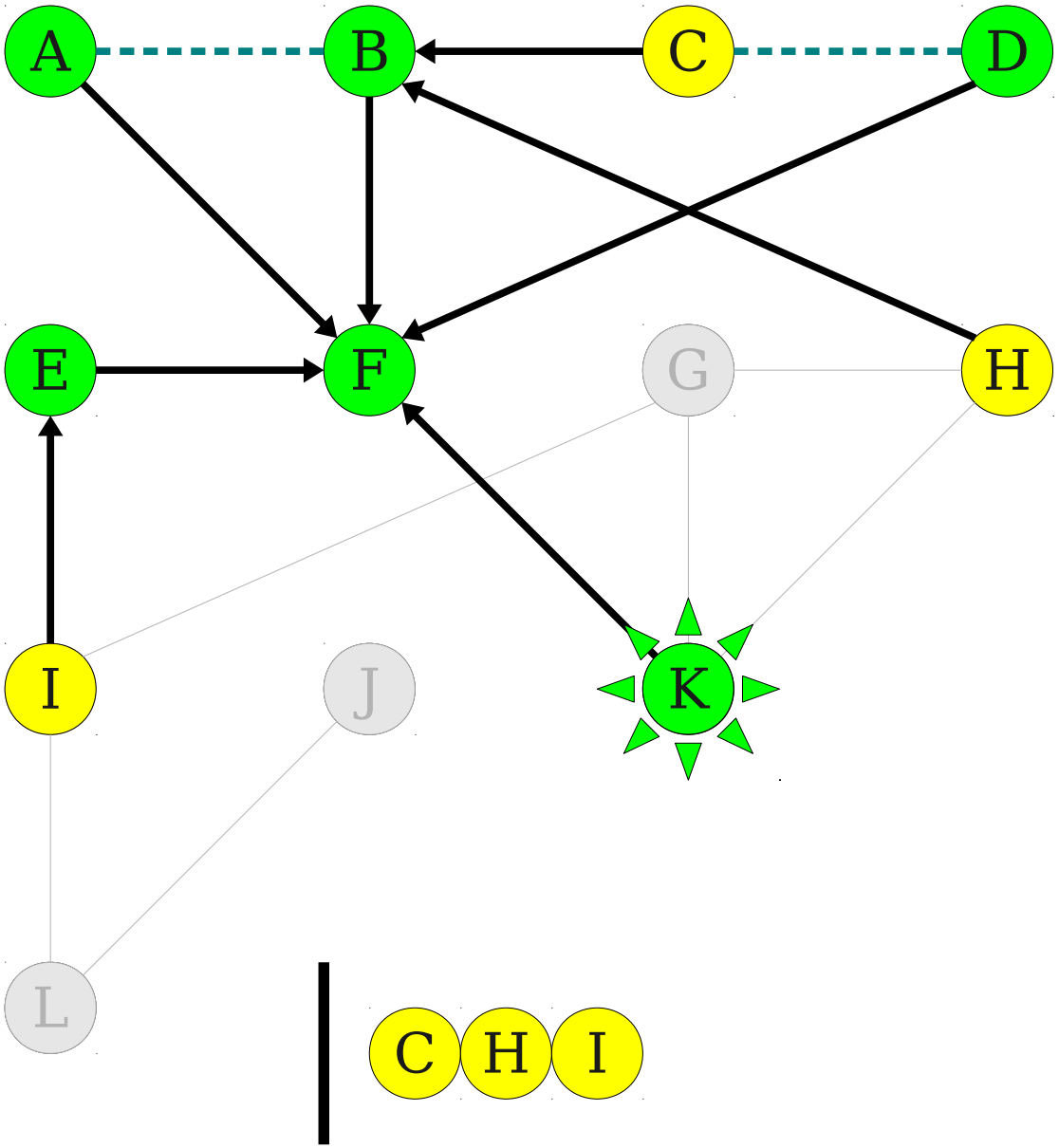
Breadth-First Search



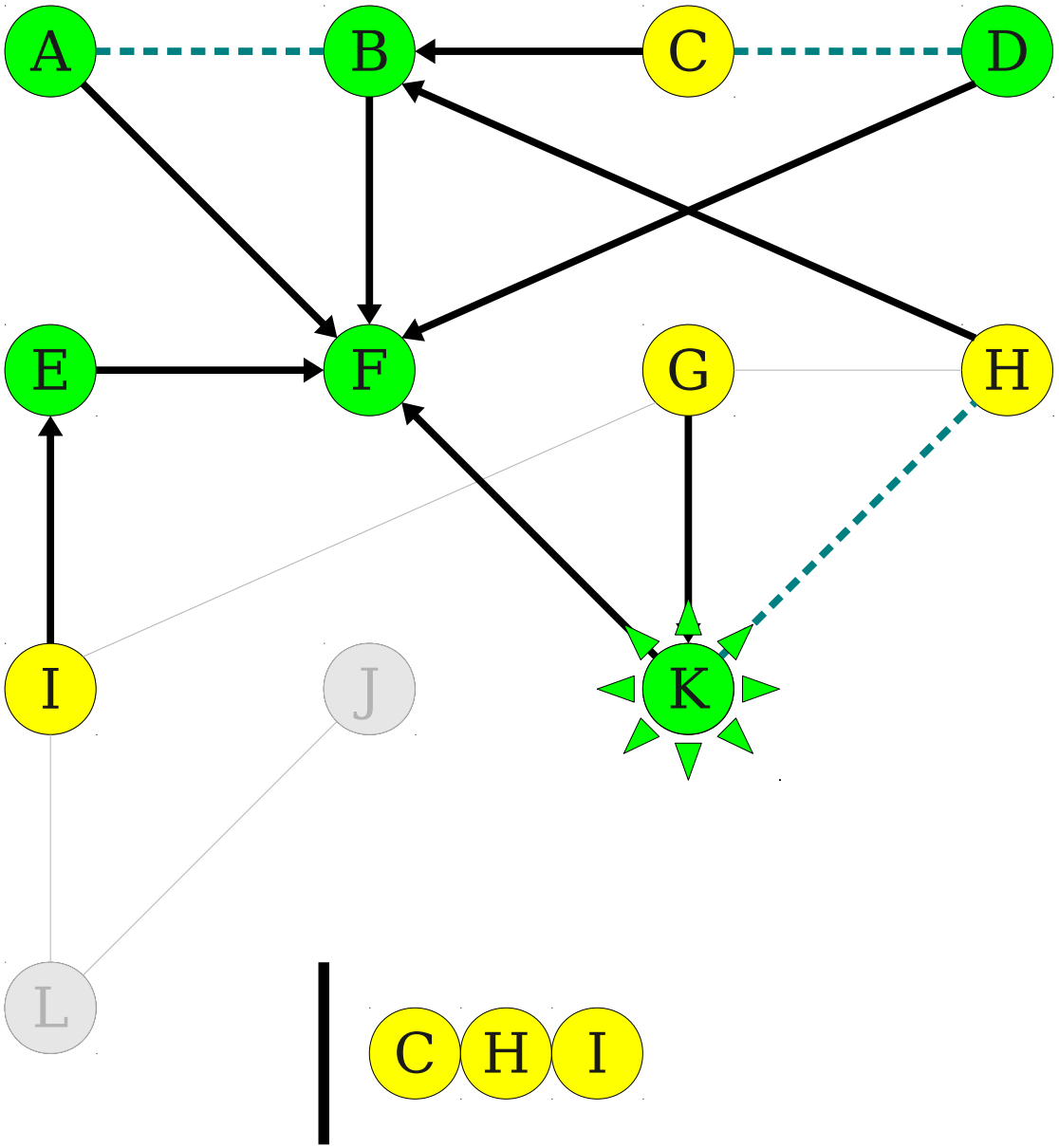
Breadth-First Search



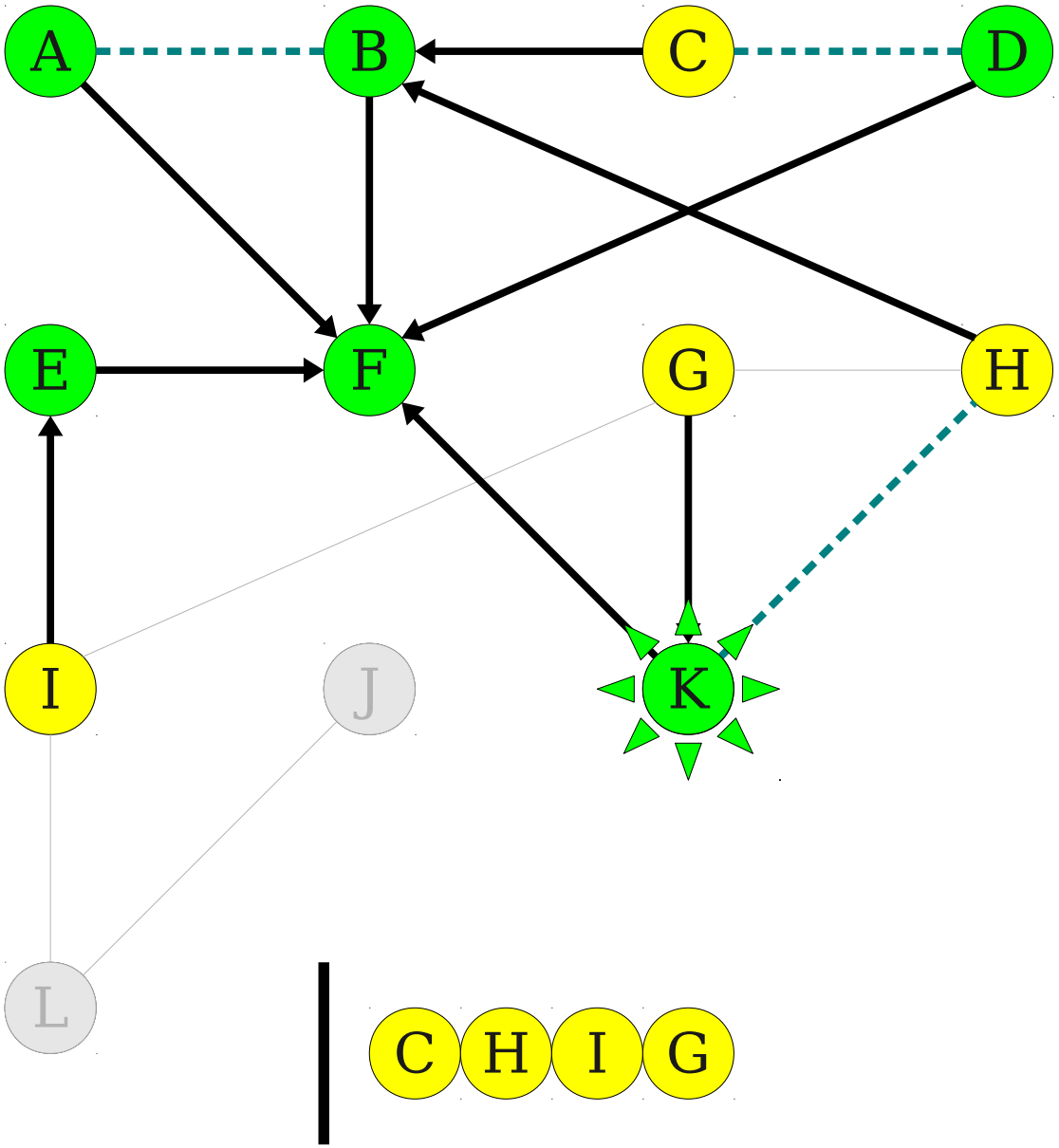
Breadth-First Search



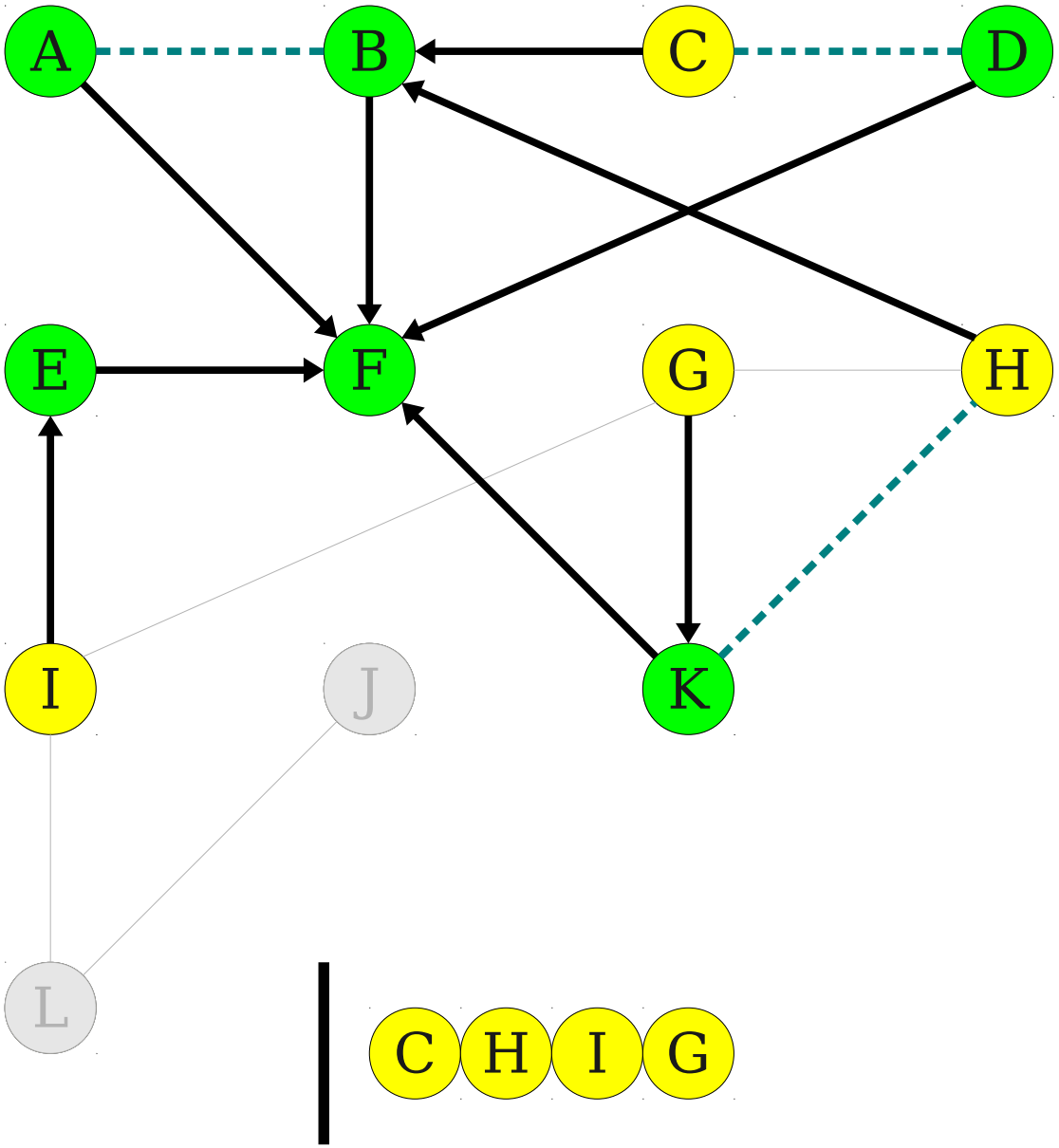
Breadth-First Search



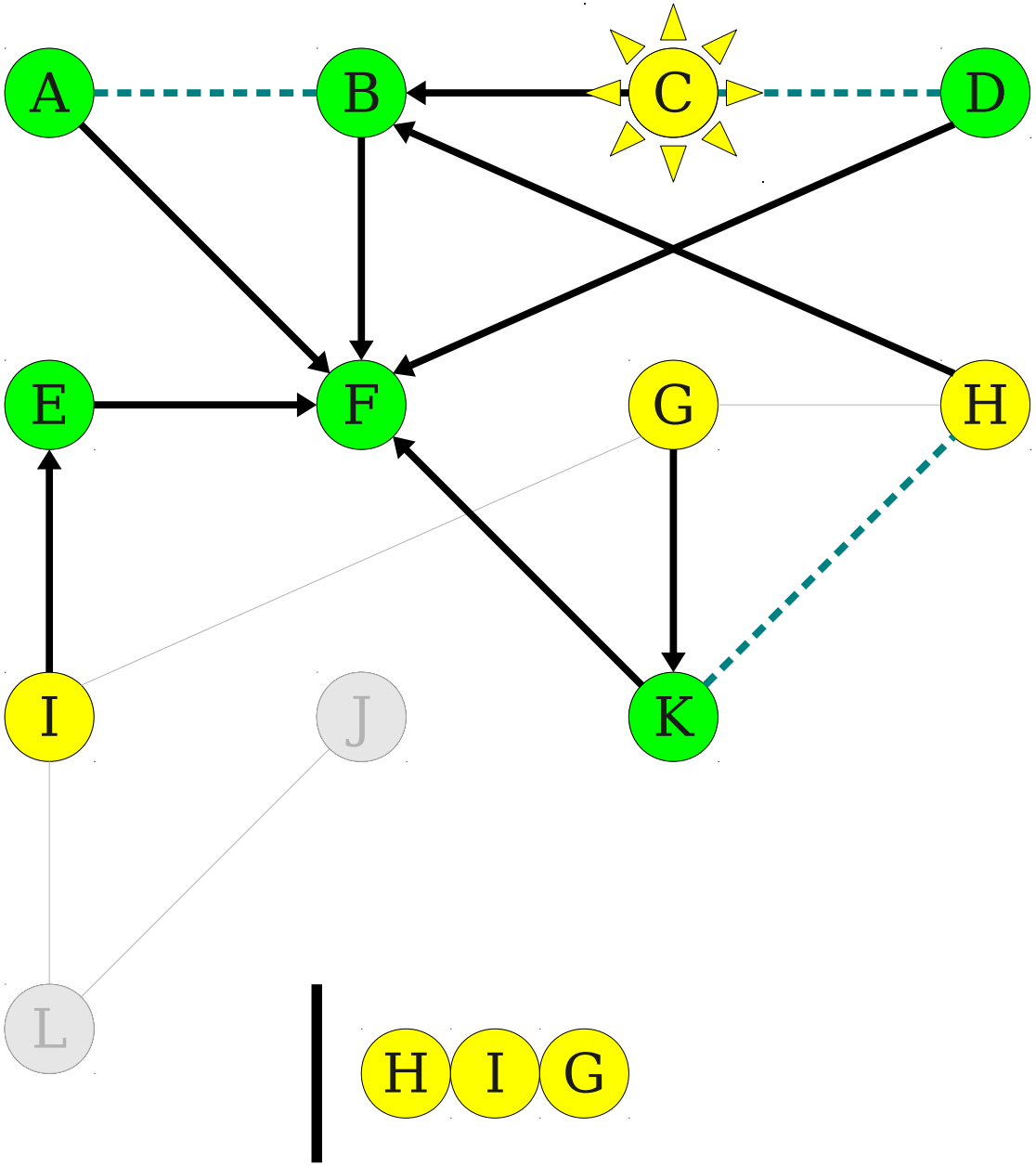
Breadth-First Search



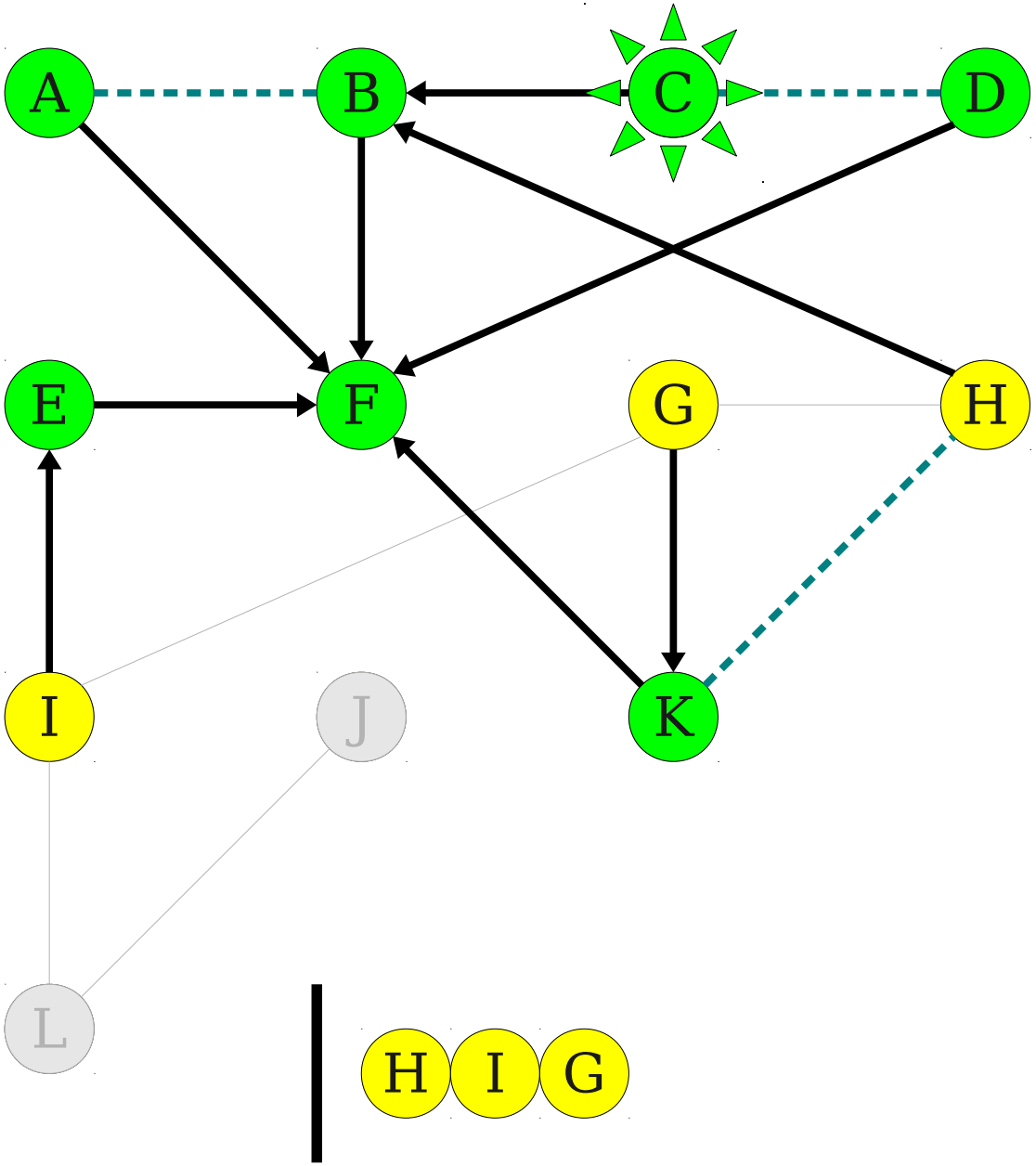
Breadth-First Search



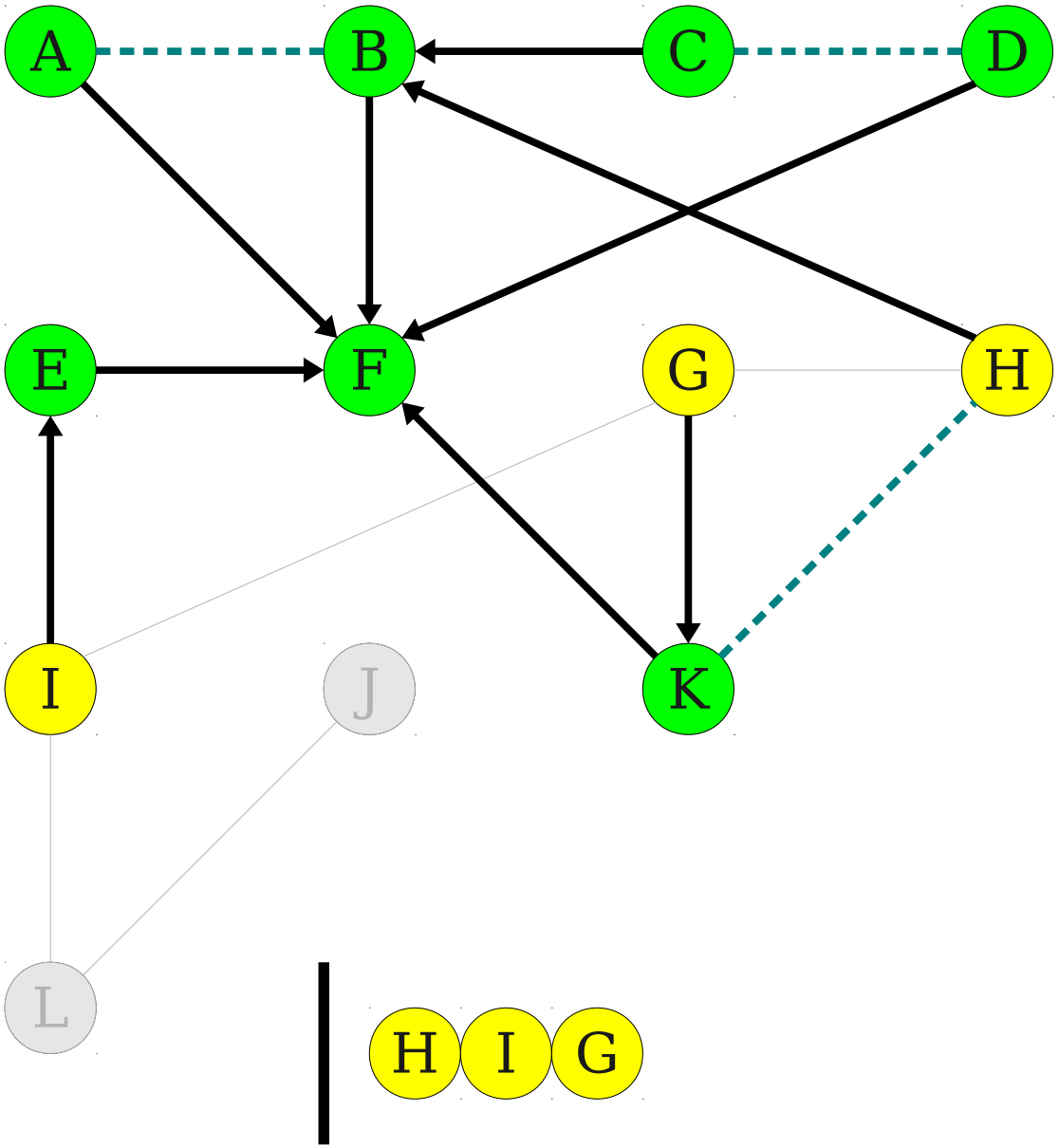
Breadth-First Search



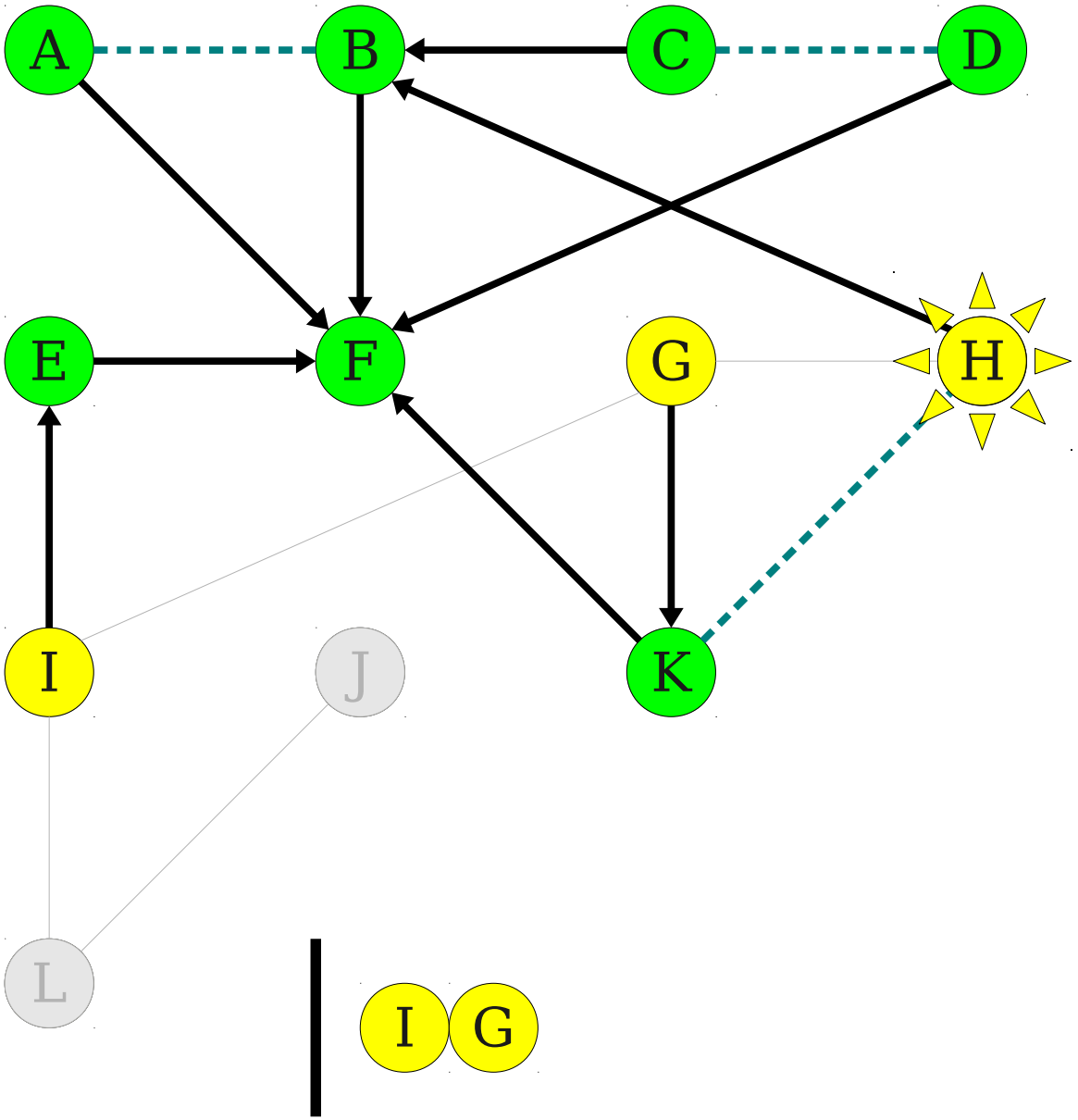
Breadth-First Search



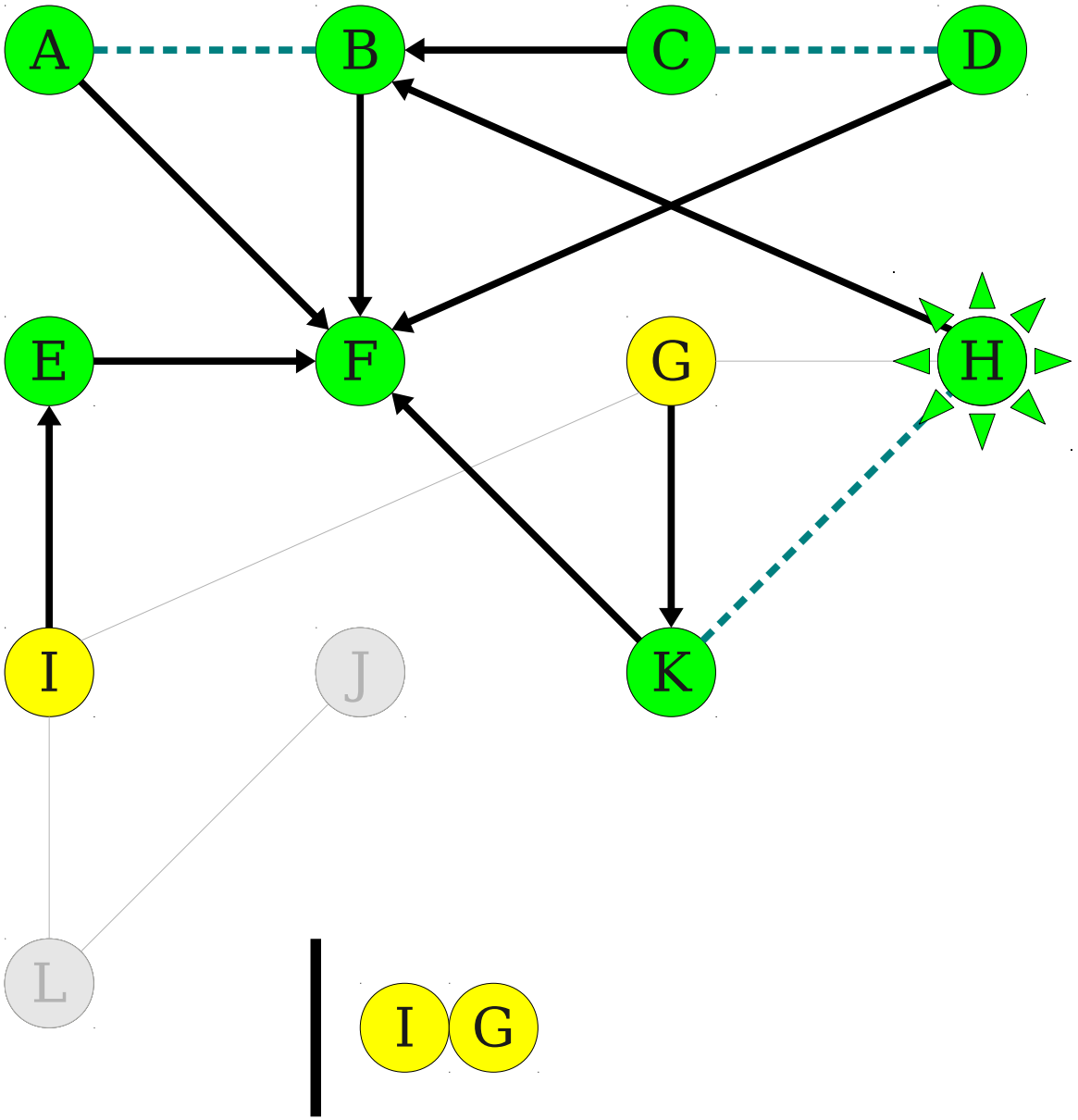
Breadth-First Search



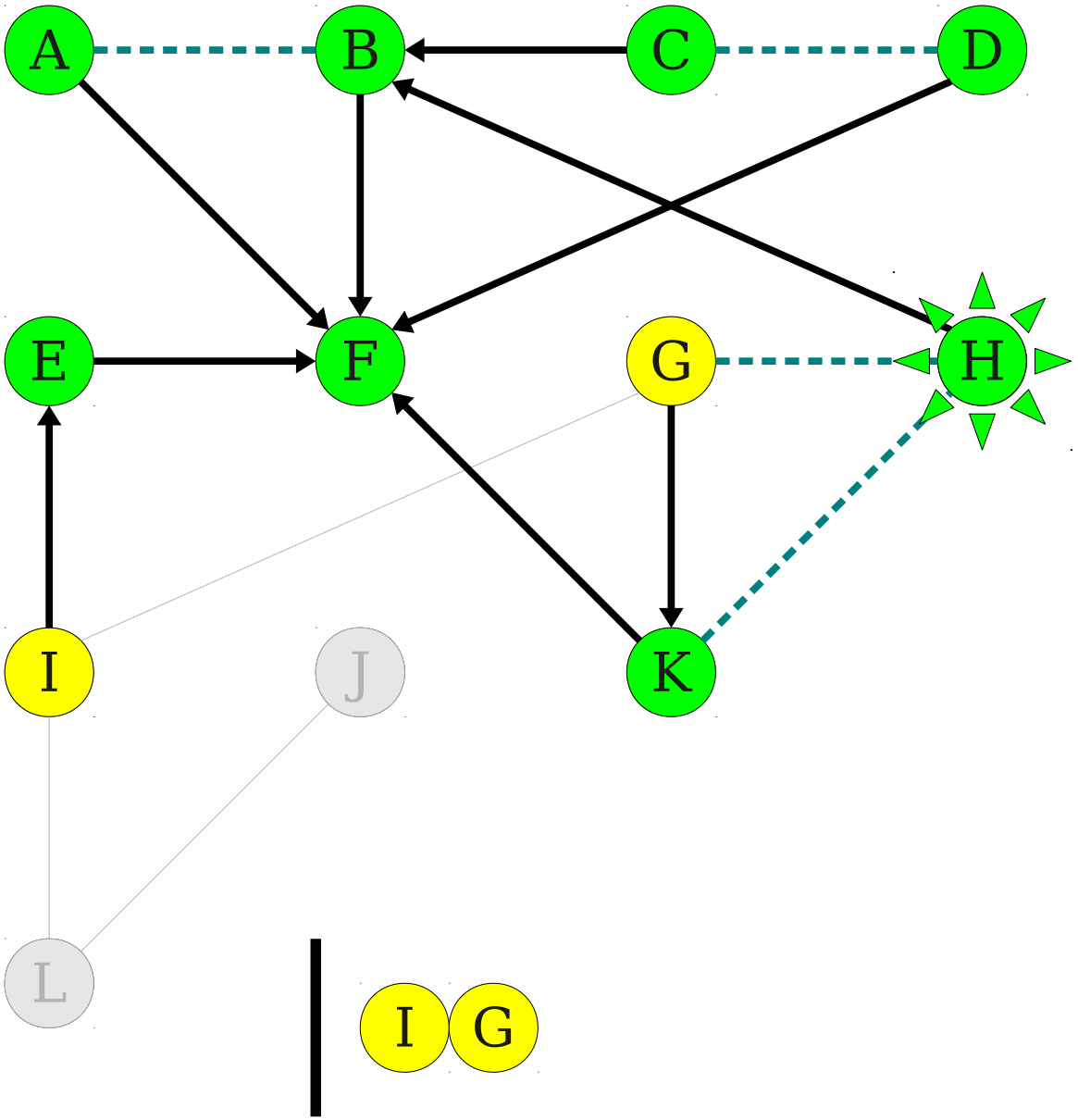
Breadth-First Search



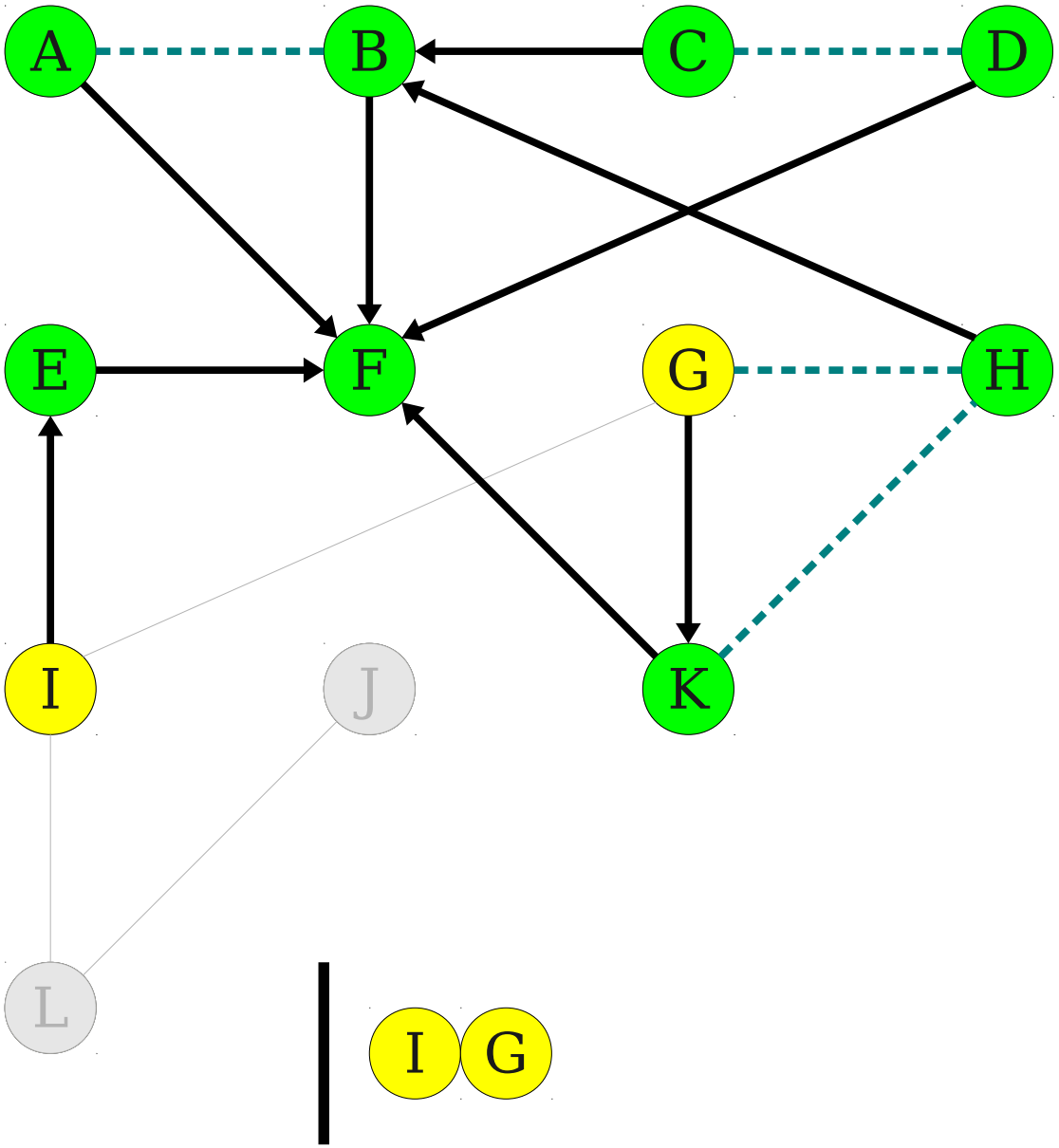
Breadth-First Search



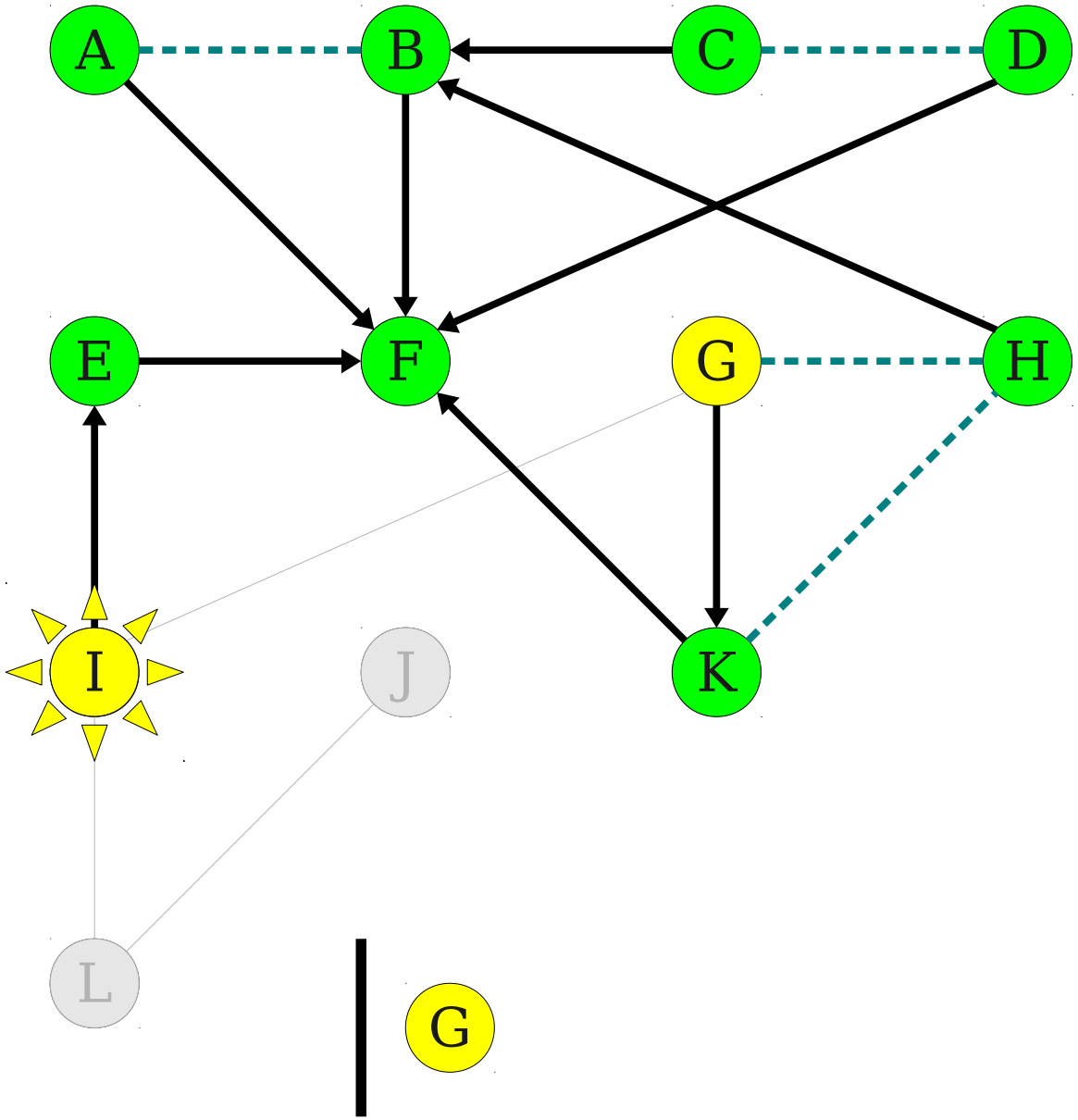
Breadth-First Search



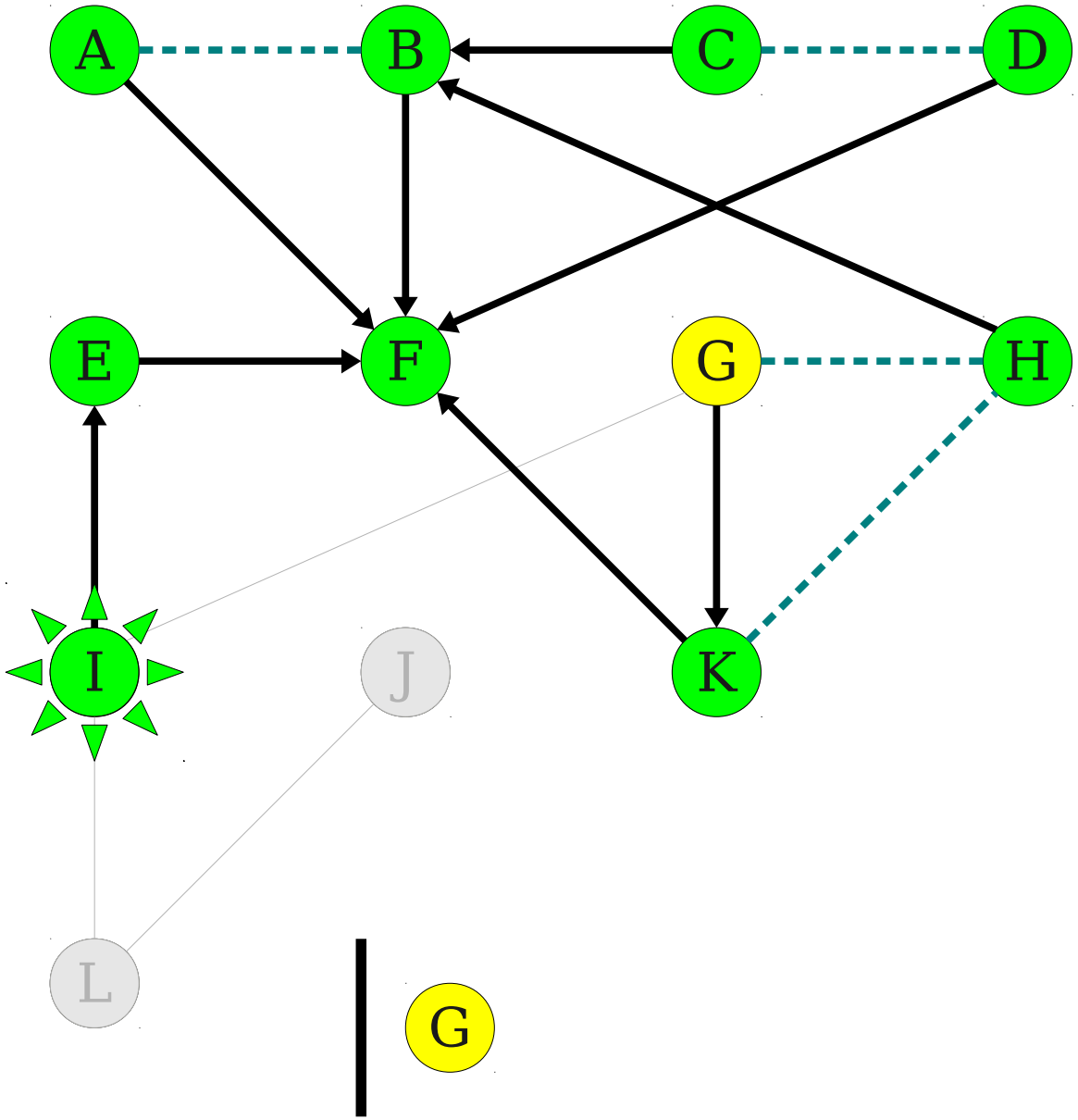
Breadth-First Search



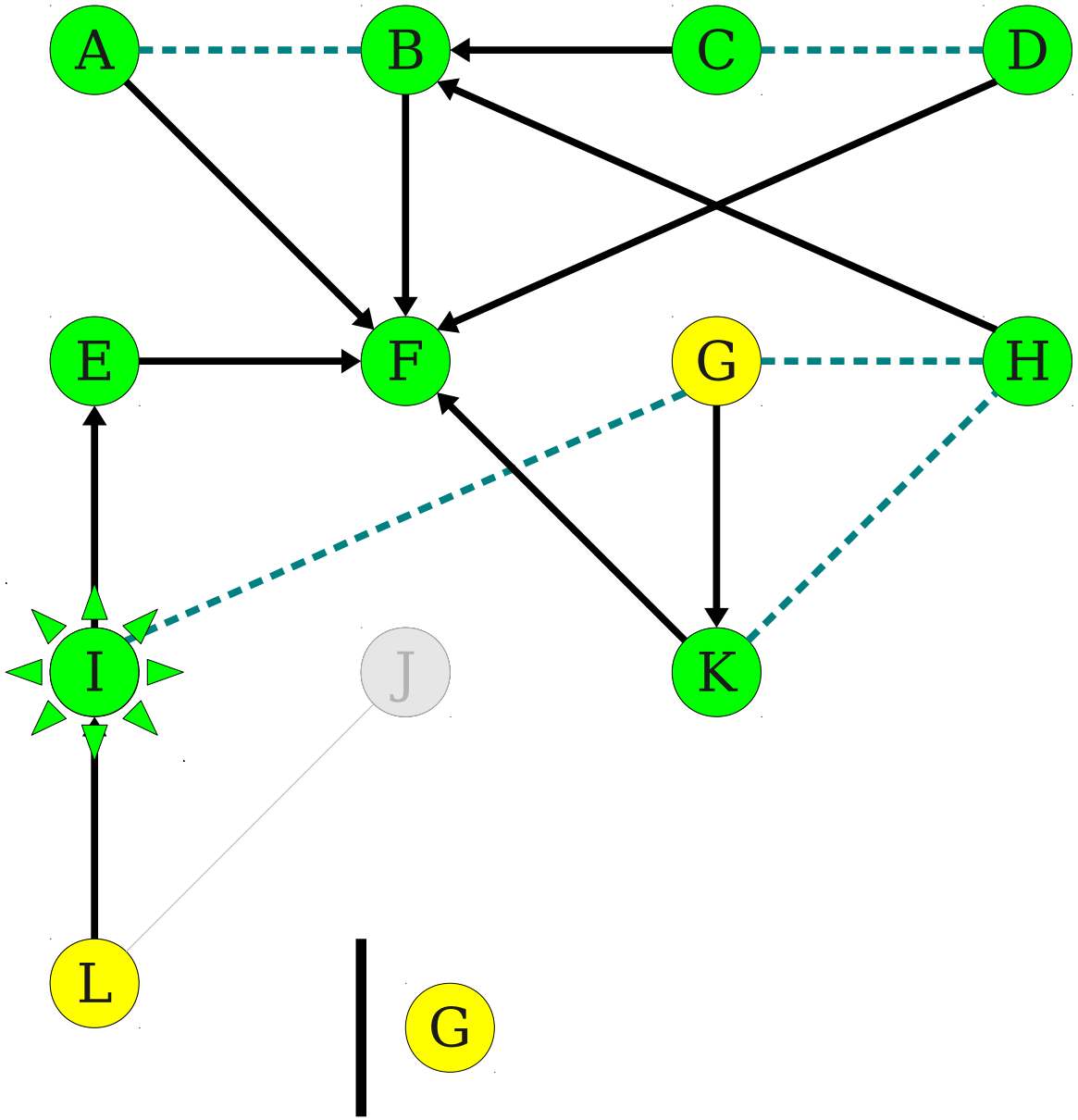
Breadth-First Search



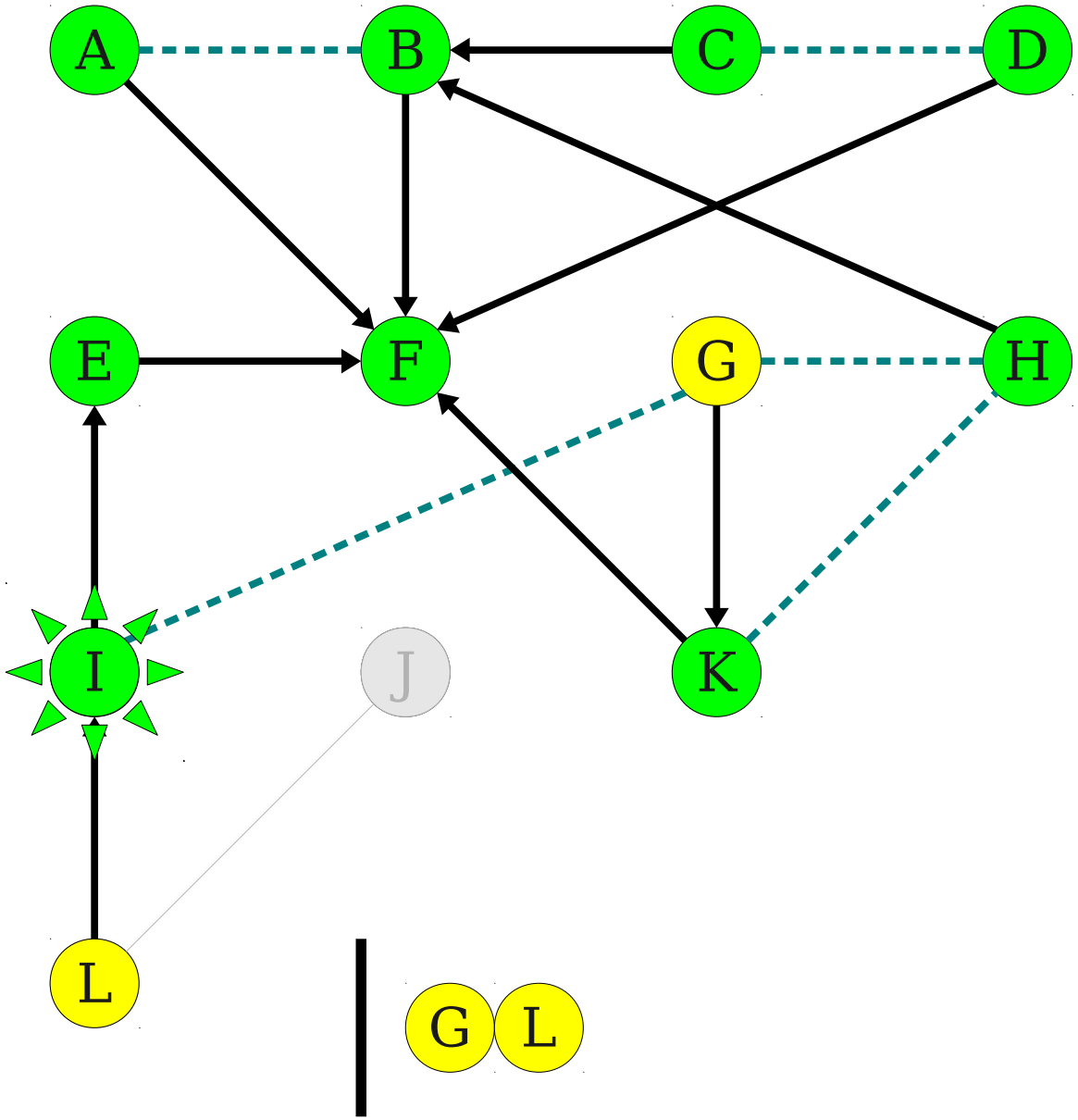
Breadth-First Search



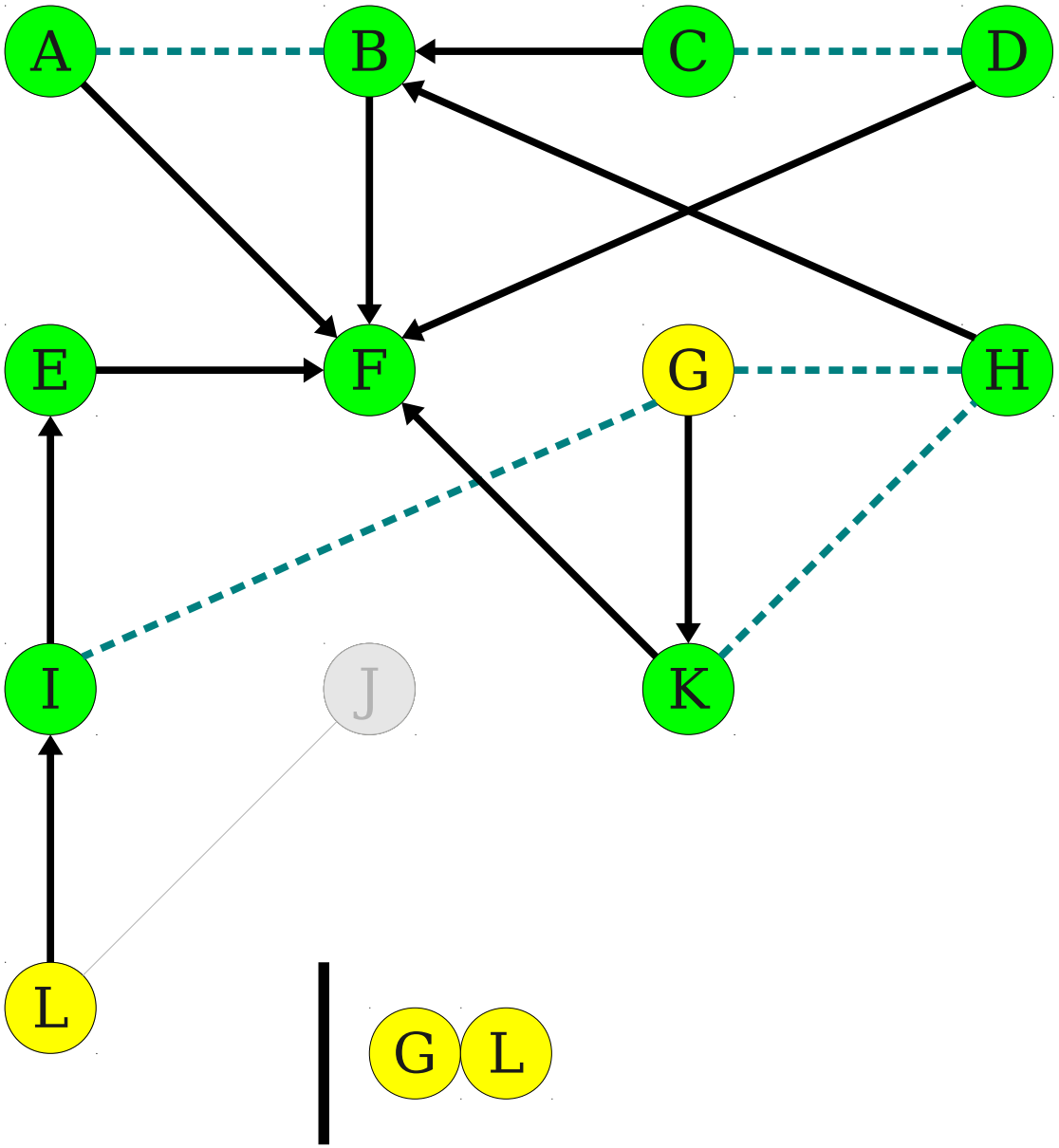
Breadth-First Search



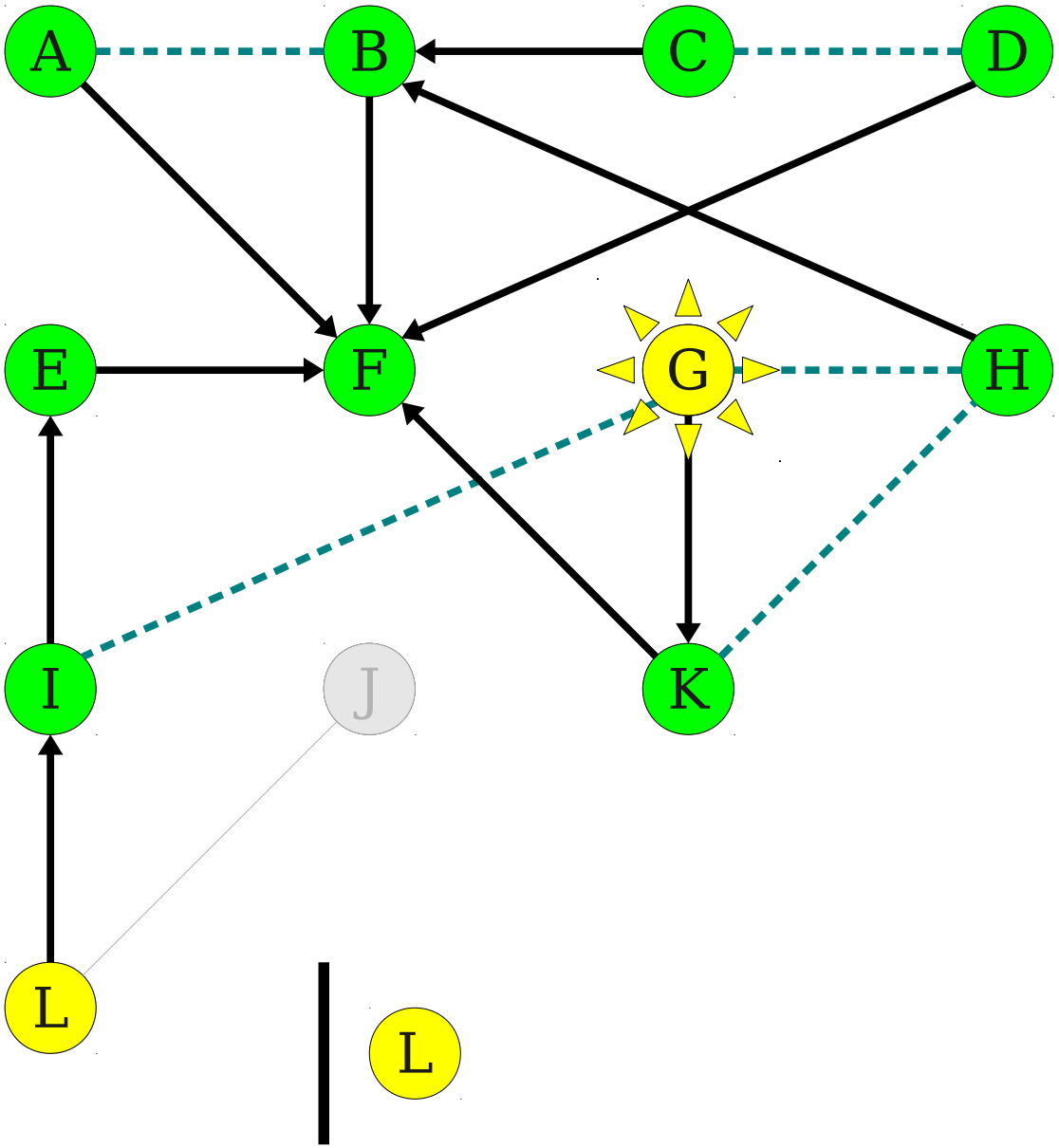
Breadth-First Search



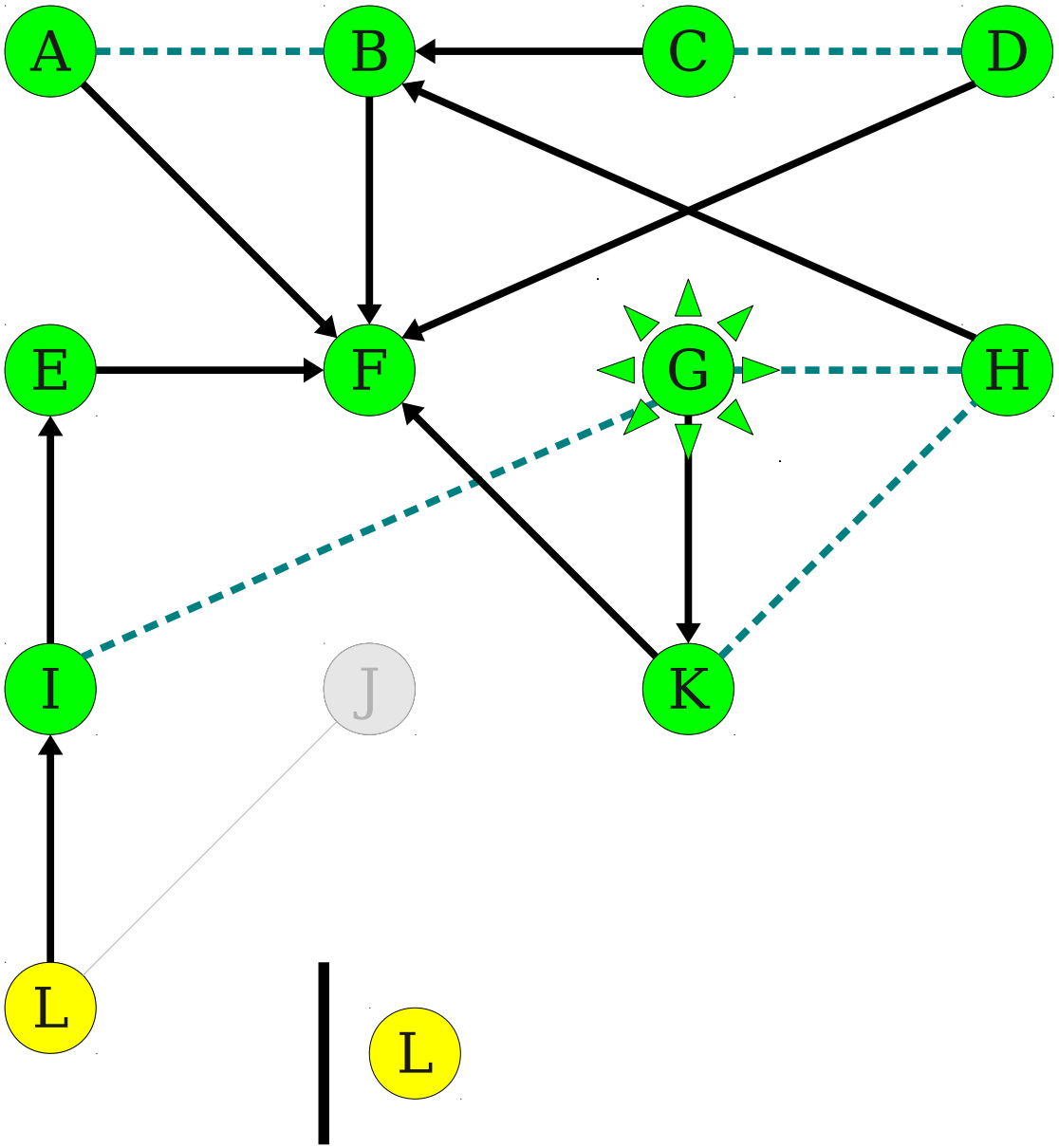
Breadth-First Search



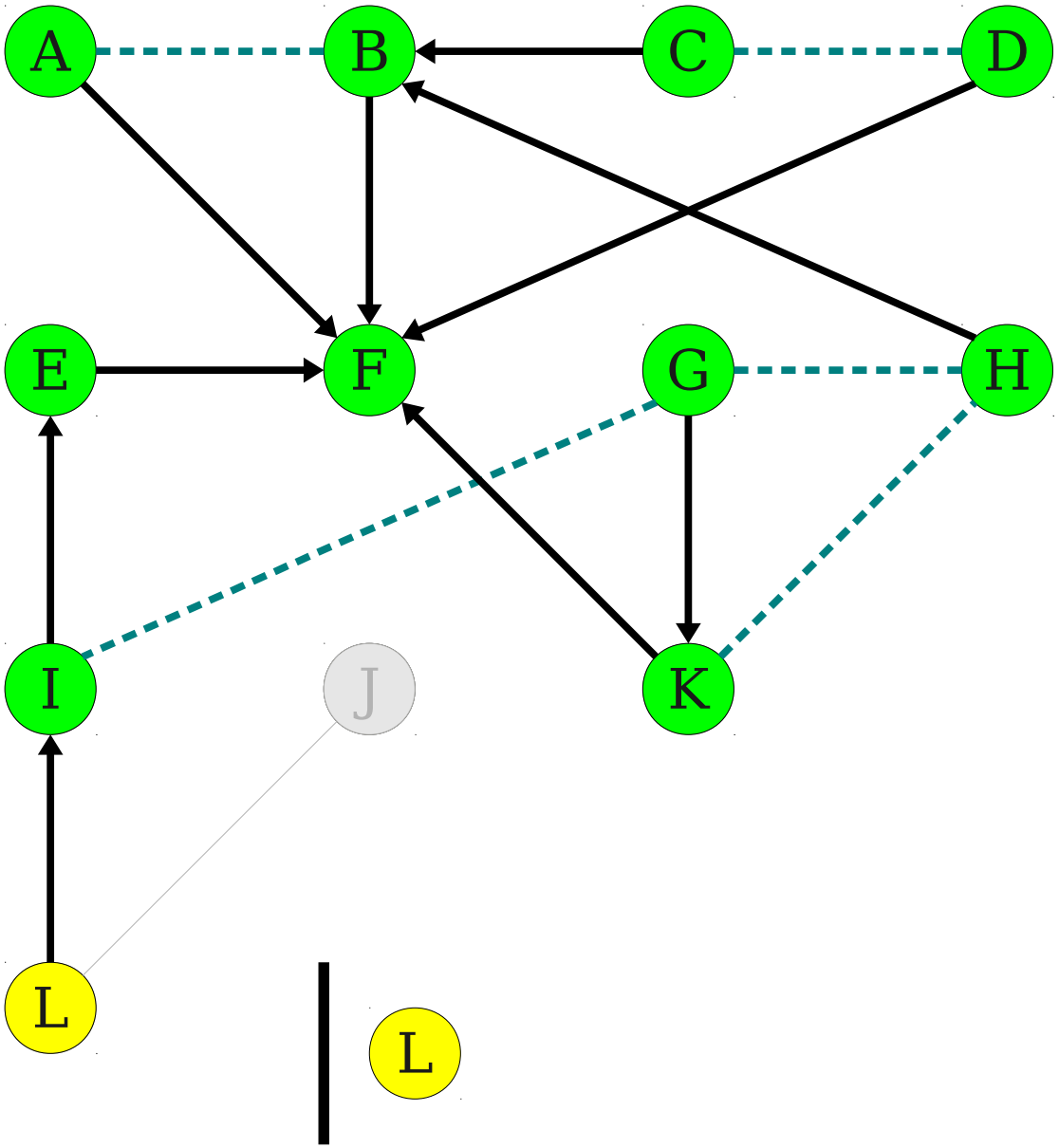
Breadth-First Search



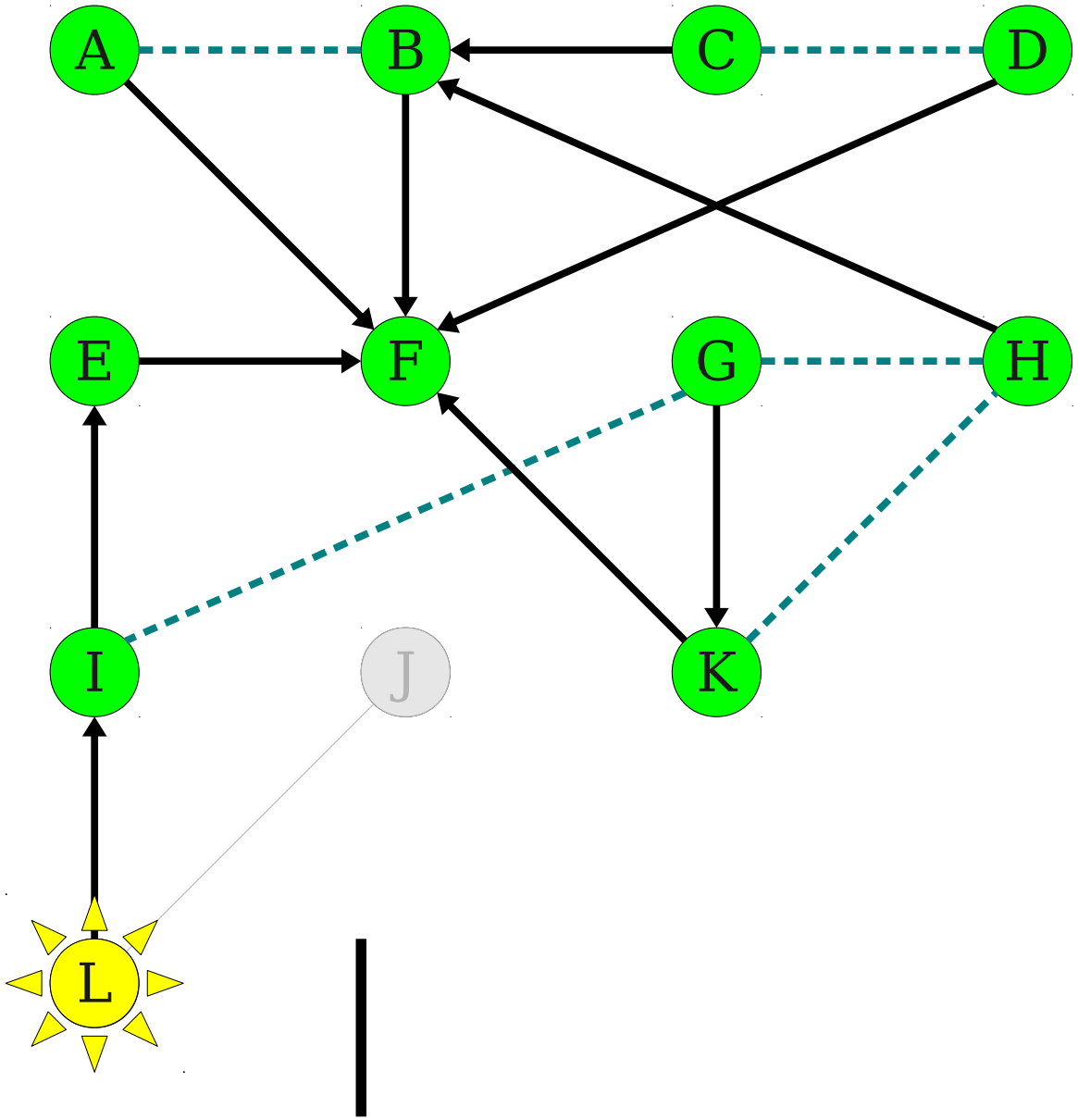
Breadth-First Search



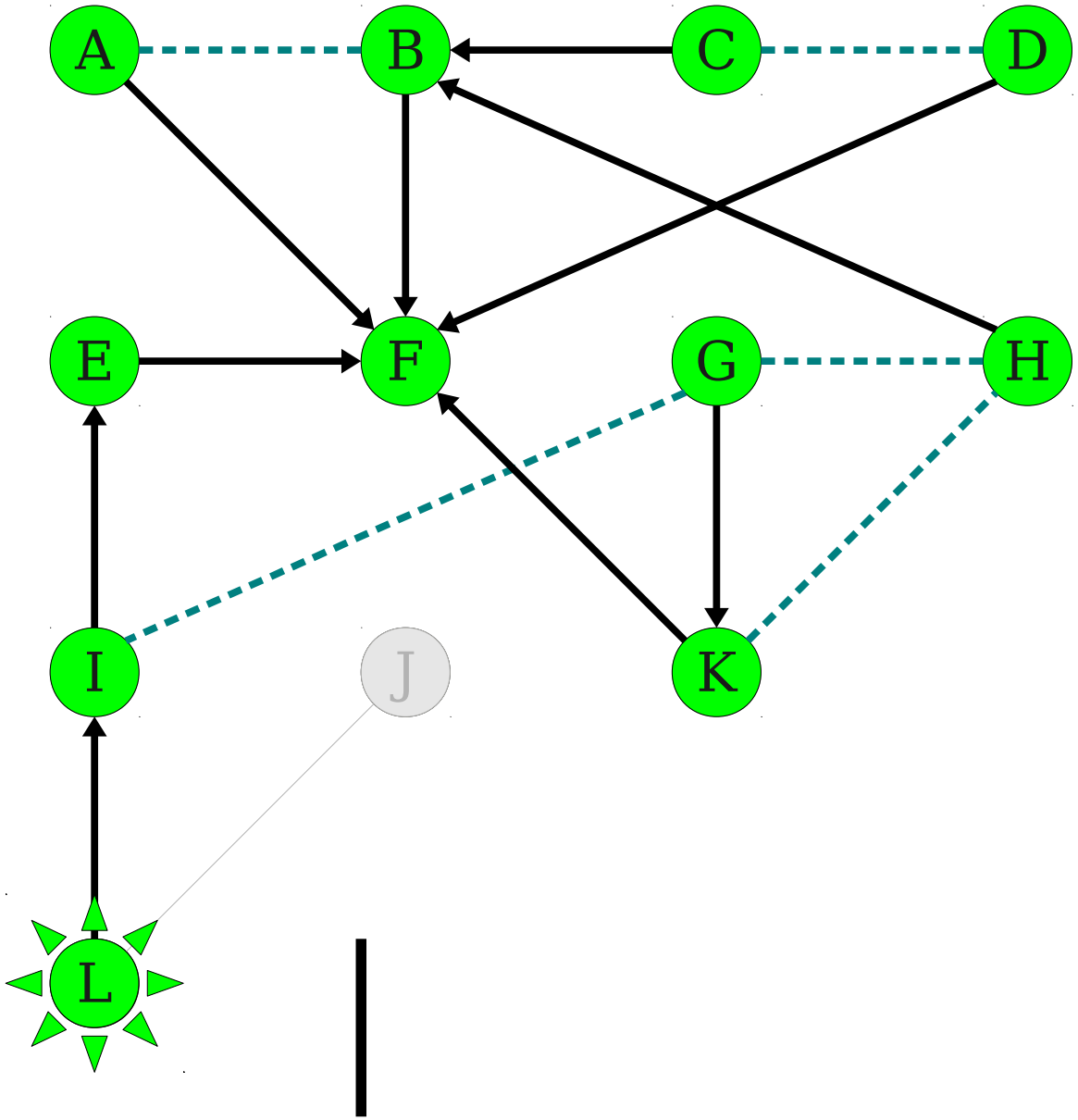
Breadth-First Search



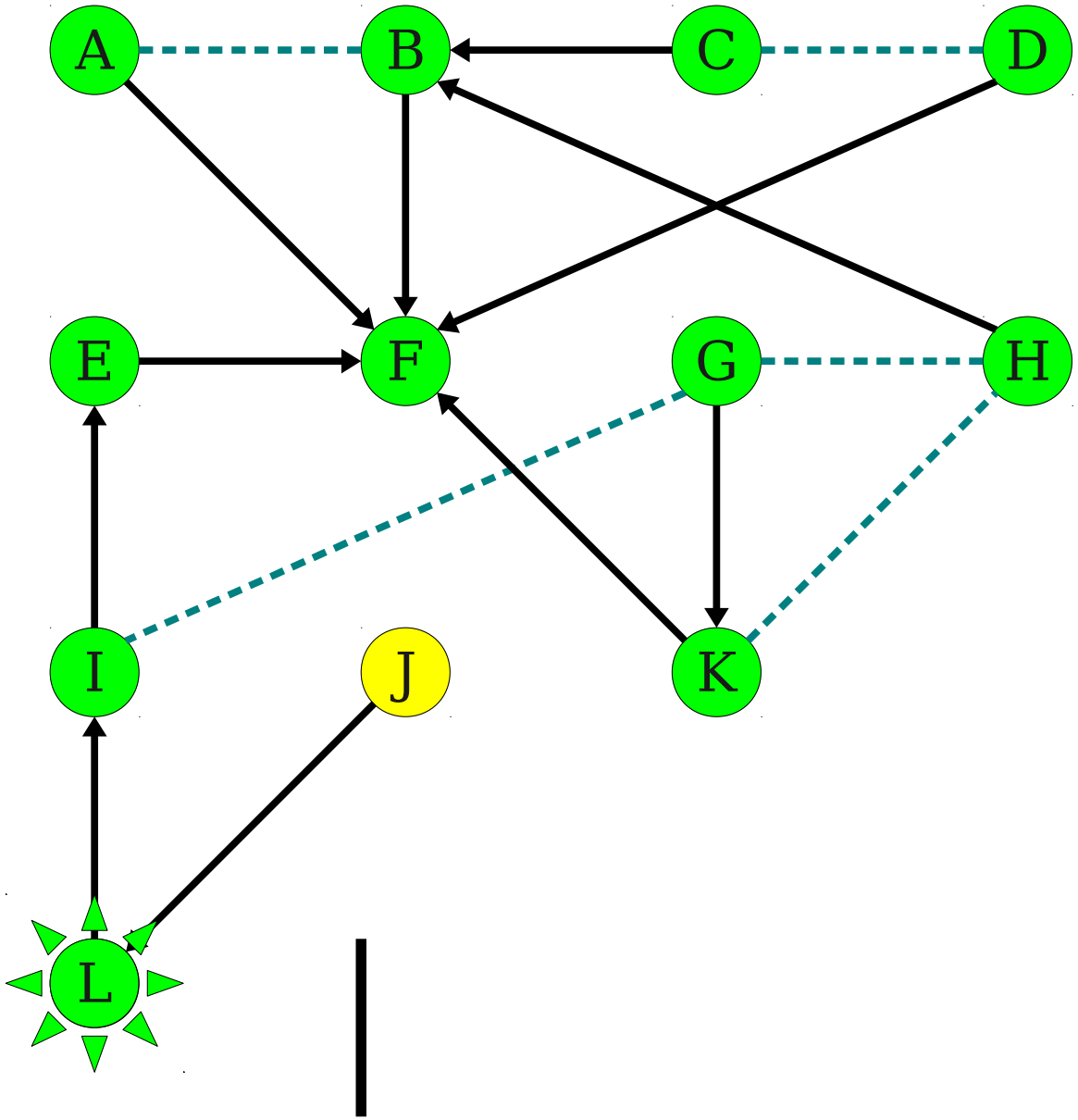
Breadth-First Search



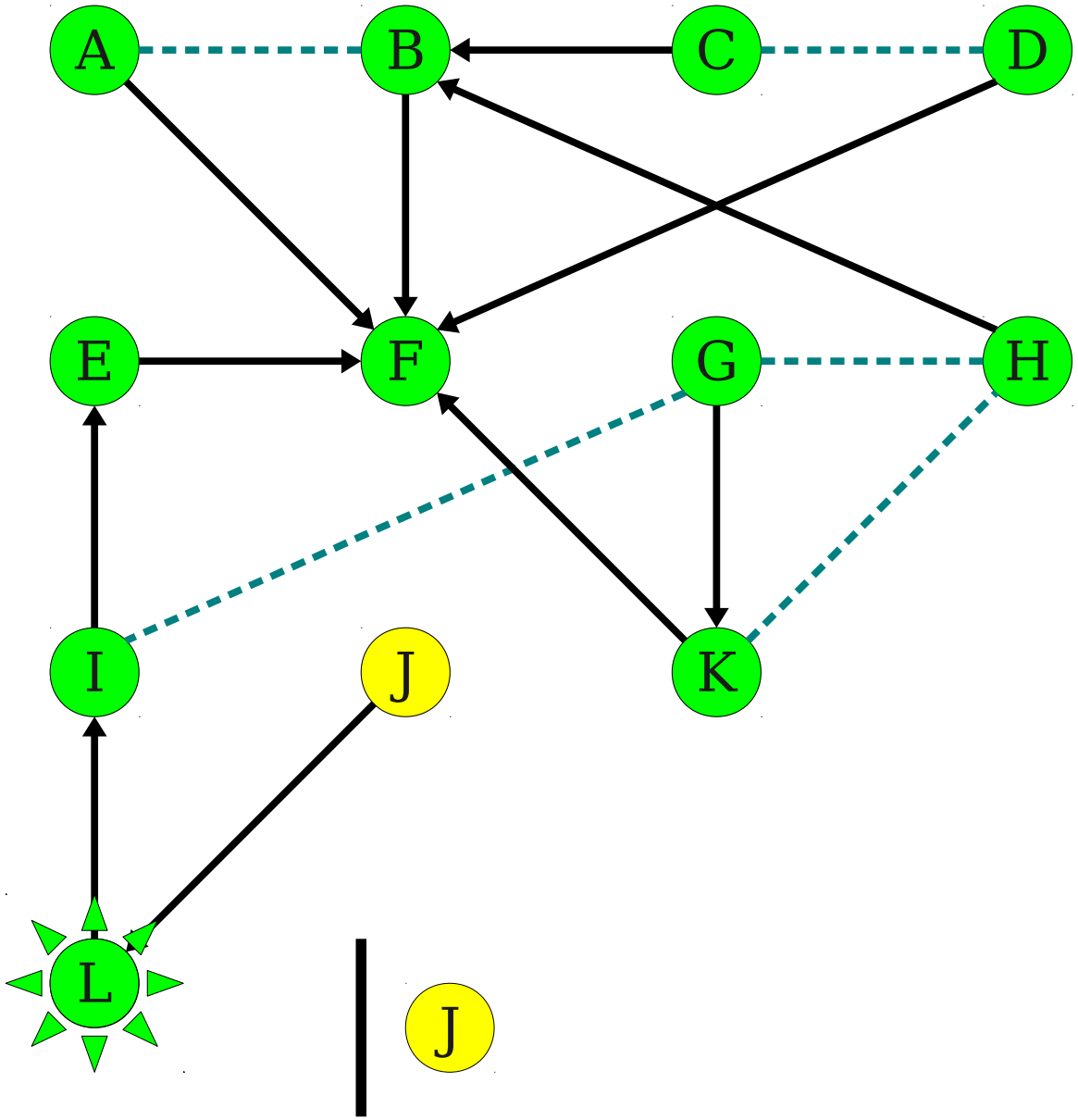
Breadth-First Search



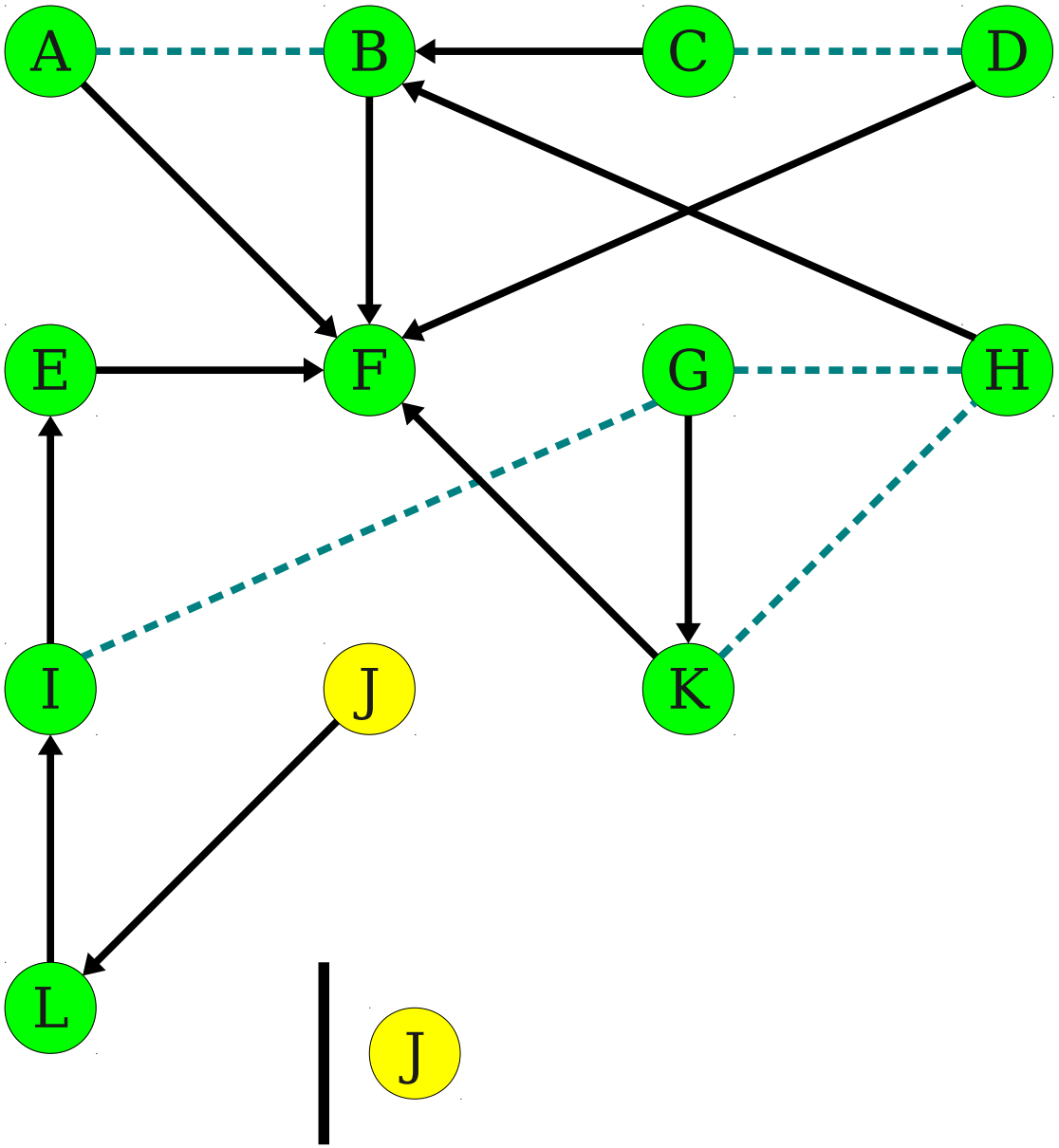
Breadth-First Search



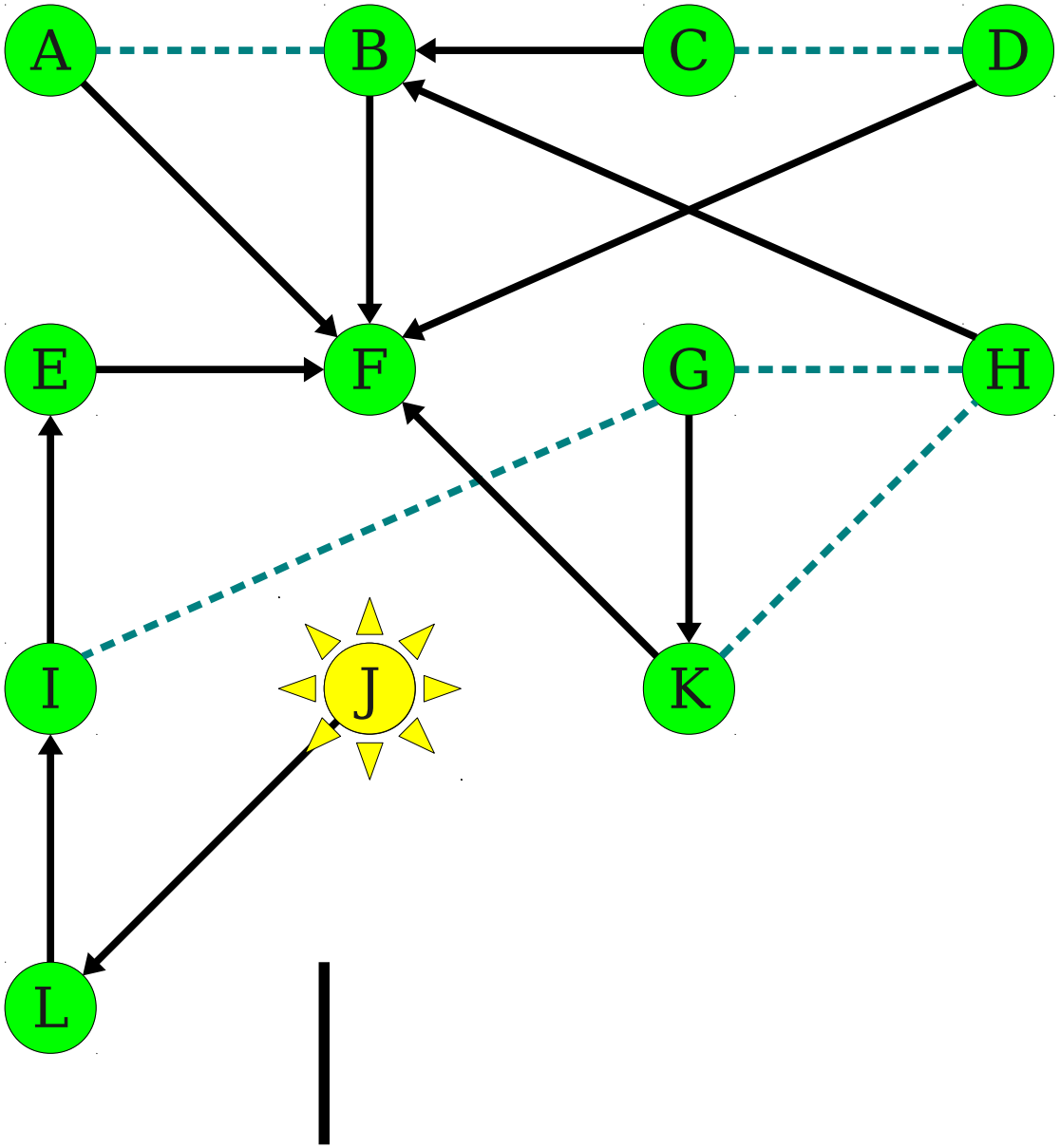
Breadth-First Search



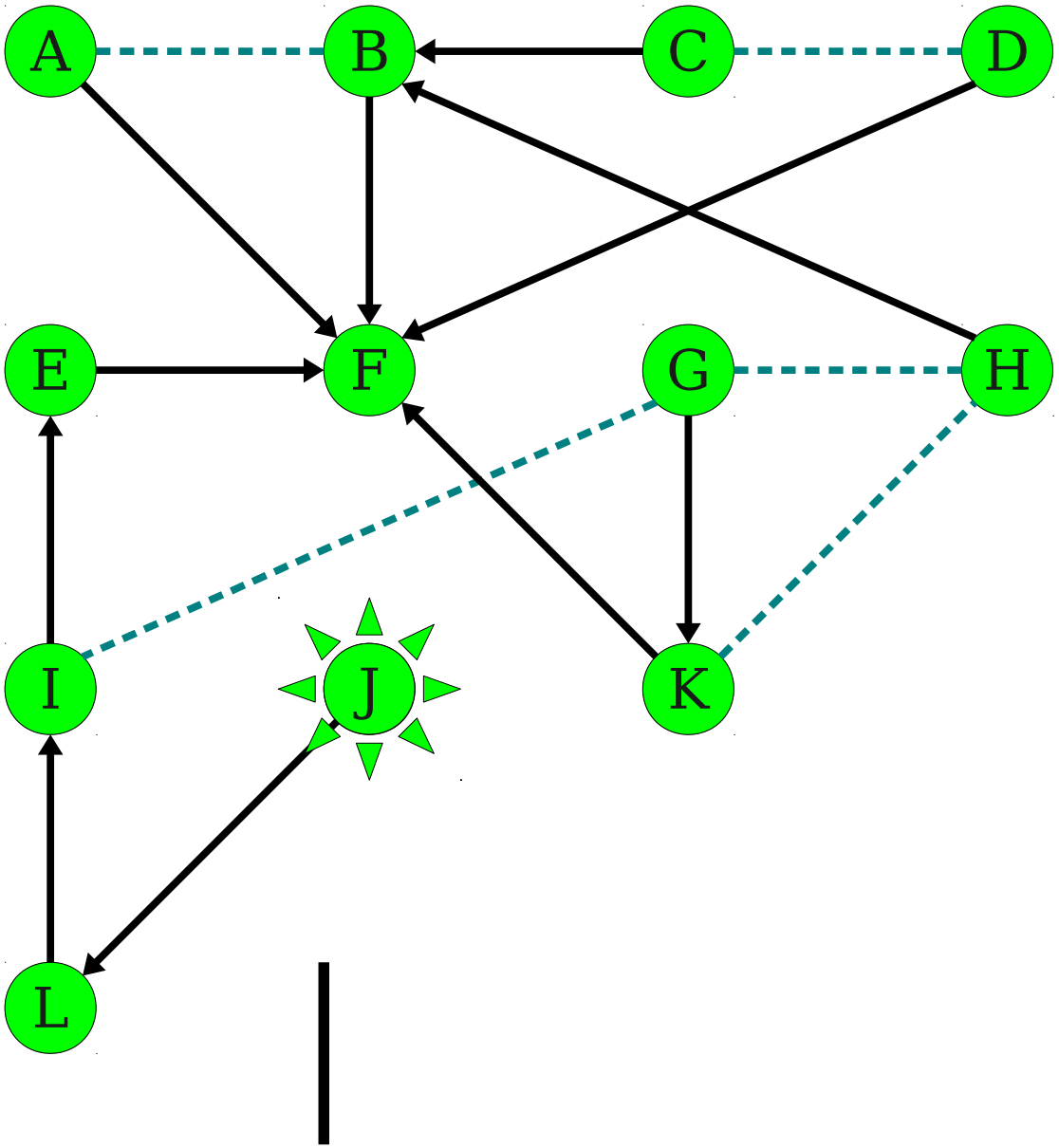
Breadth-First Search



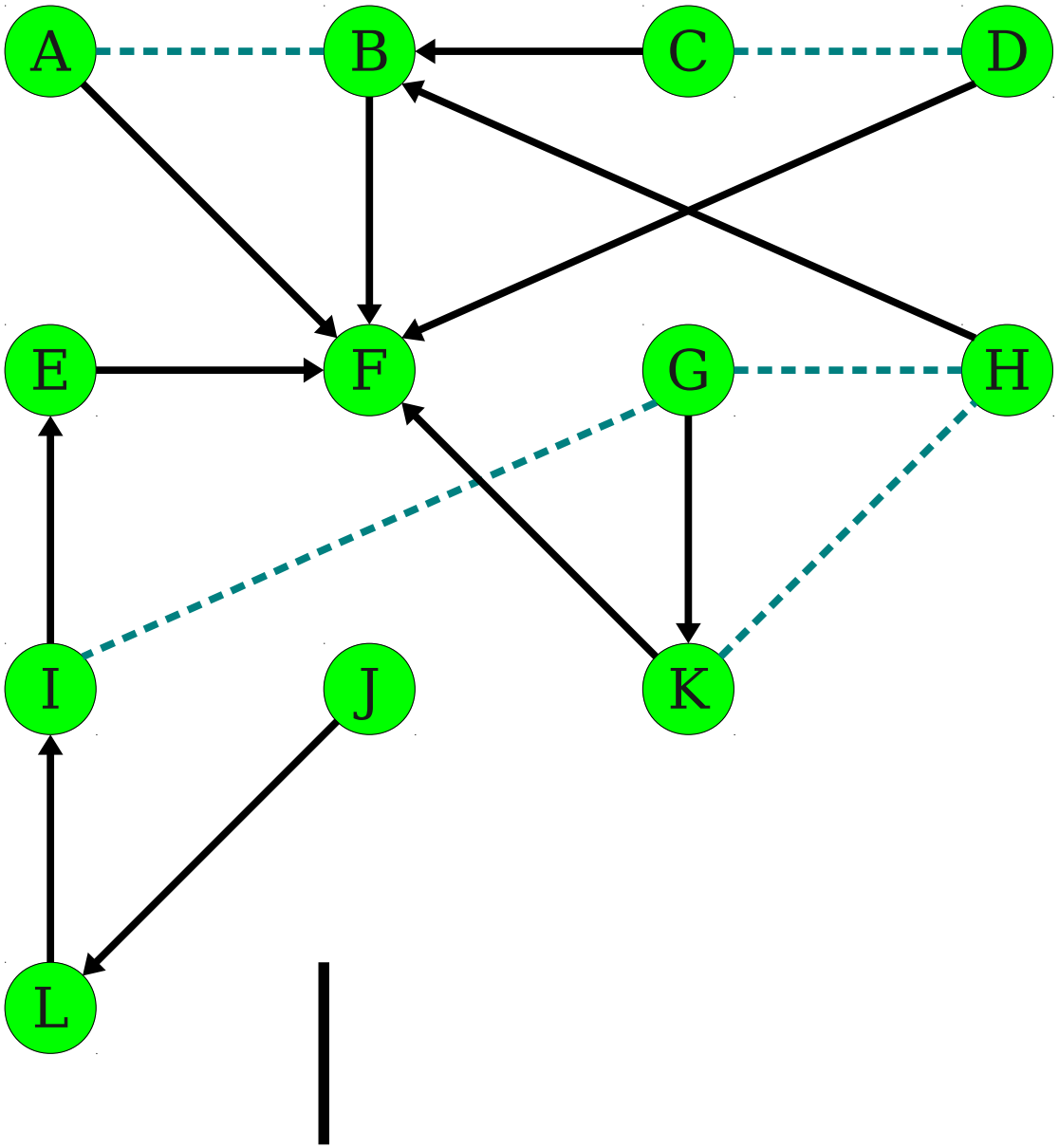
Breadth-First Search



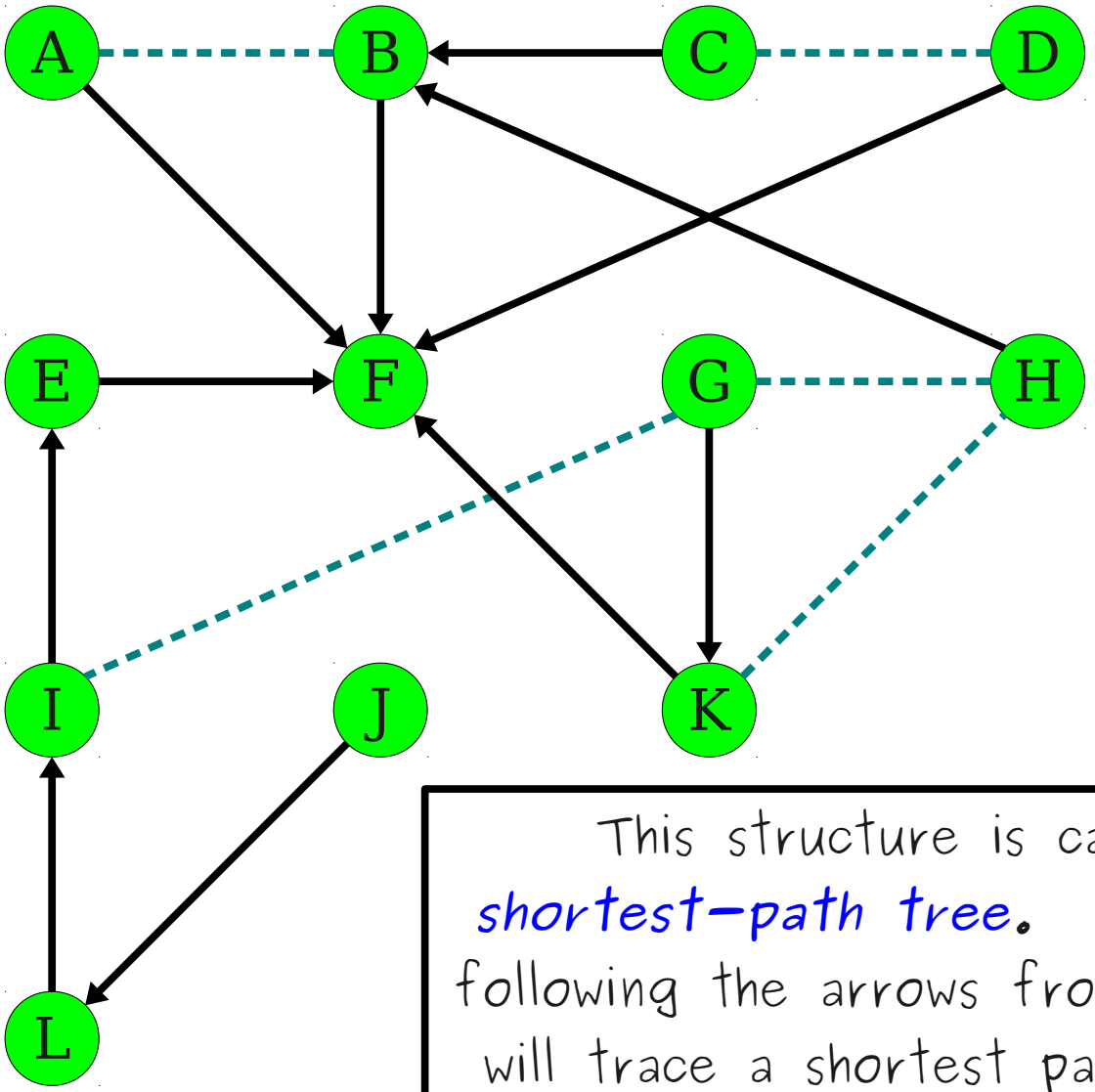
Breadth-First Search



Breadth-First Search

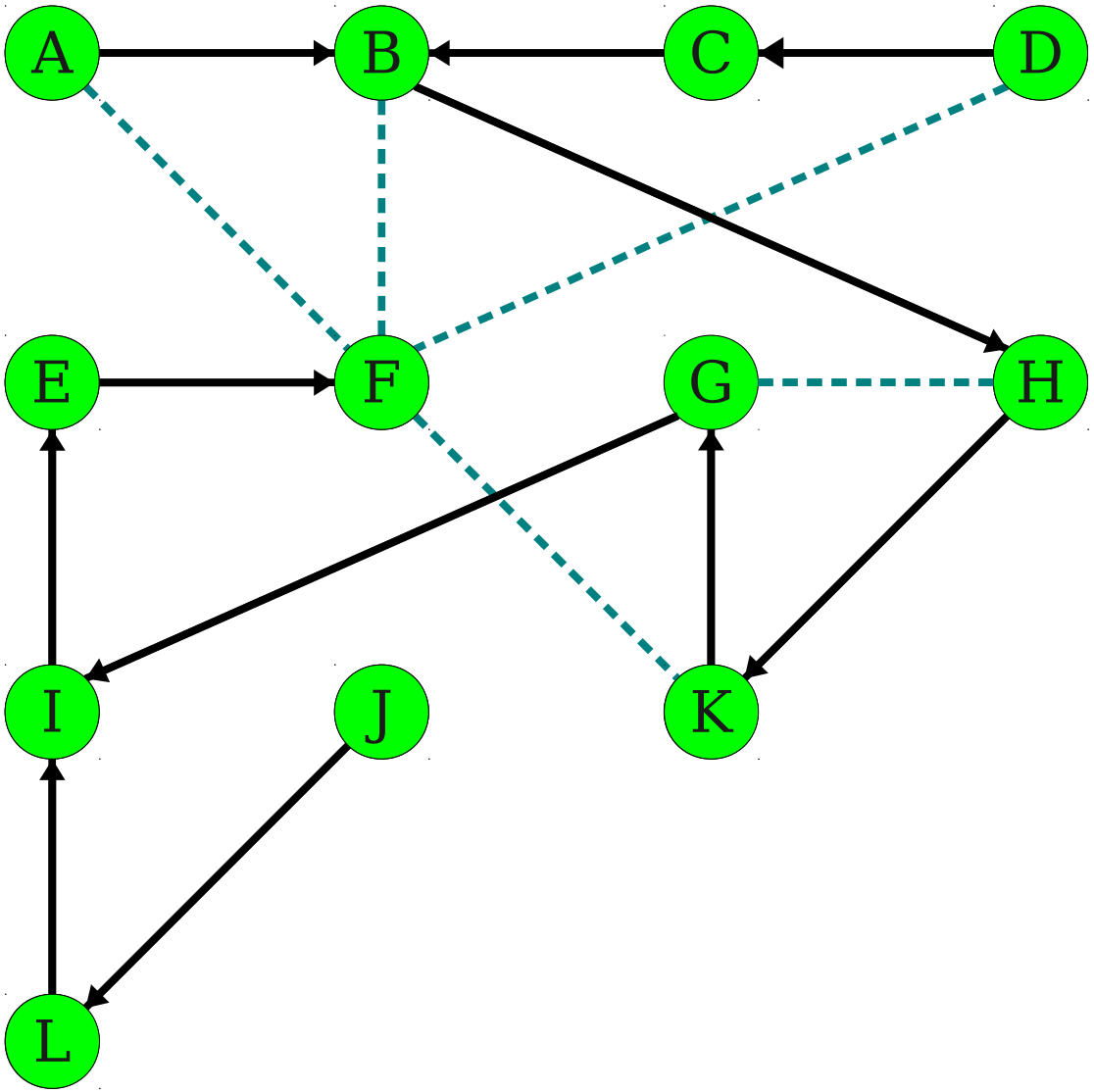


Breadth-First Search

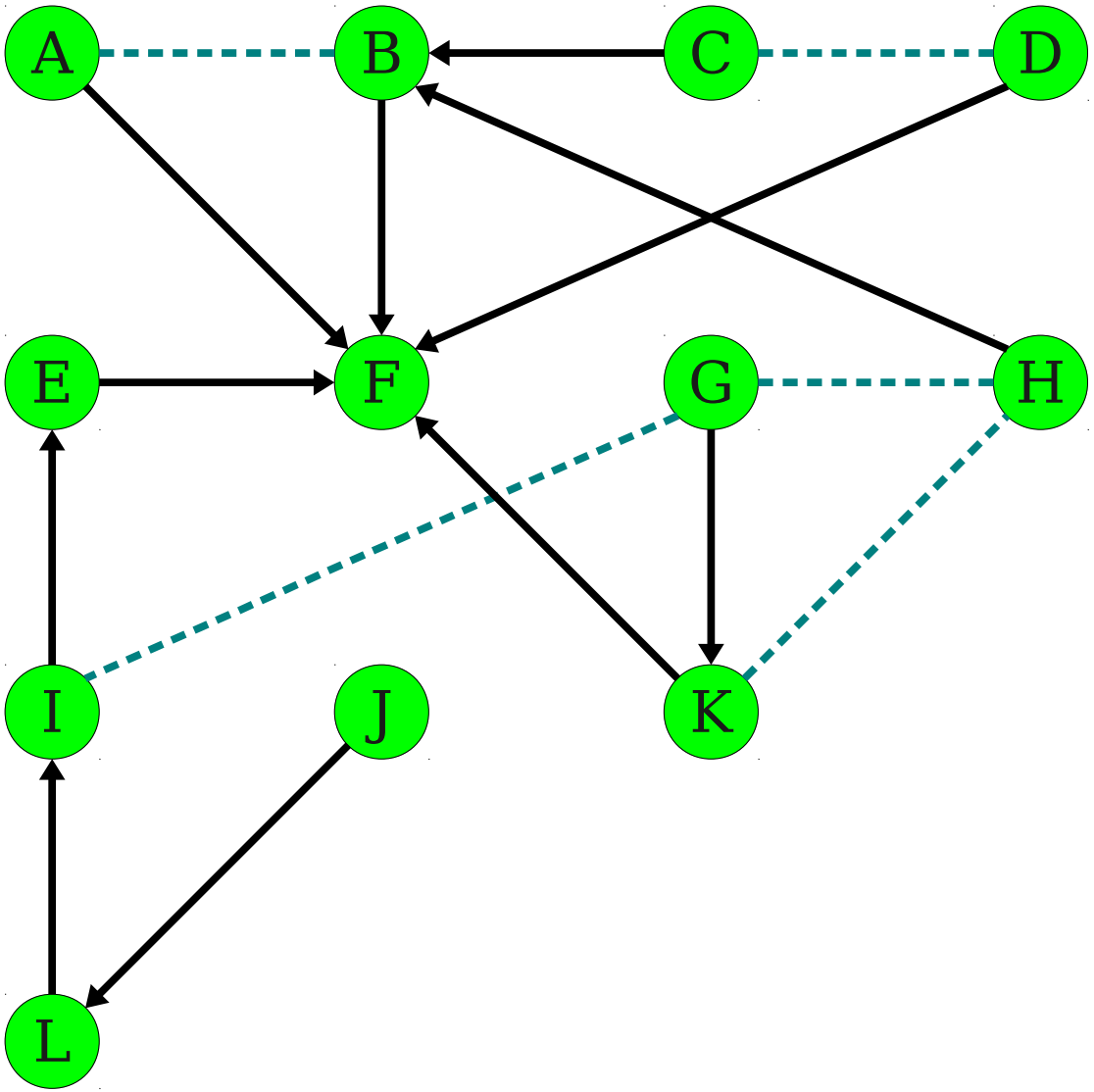


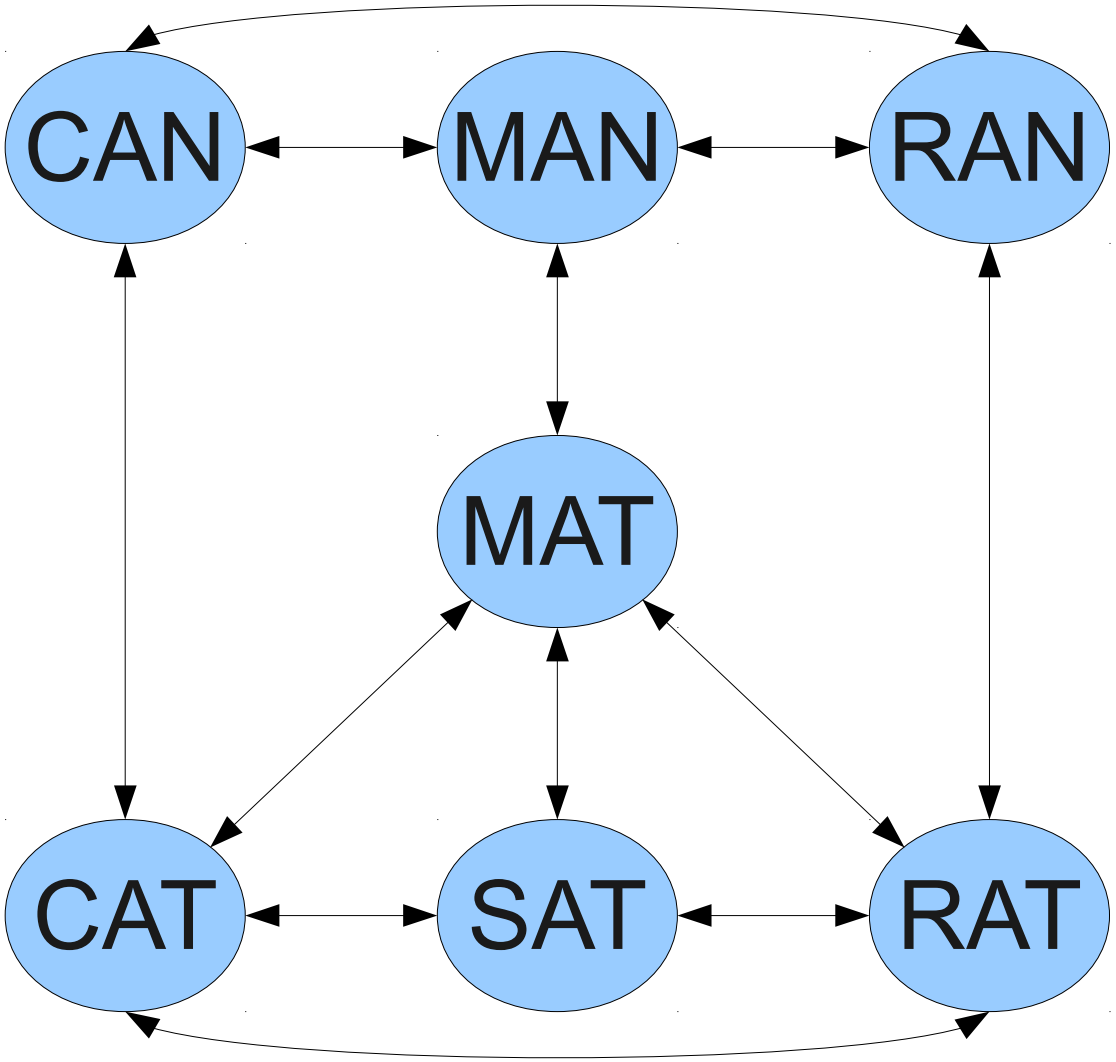
This structure is called a *shortest-path tree*. Notice how following the arrows from any node will trace a shortest path back to the root in reverse.

For Comparison: DFS



For Comparison: BFS




CAN
MAN
RAN
MAT
CAT
SAT
RAT

Next Time

- **Shortest Paths**
  - Dijkstra's Algorithm.
  - Shortest-Path Trees
  - A* Search.
  - ITA: Minimum Spanning Trees I