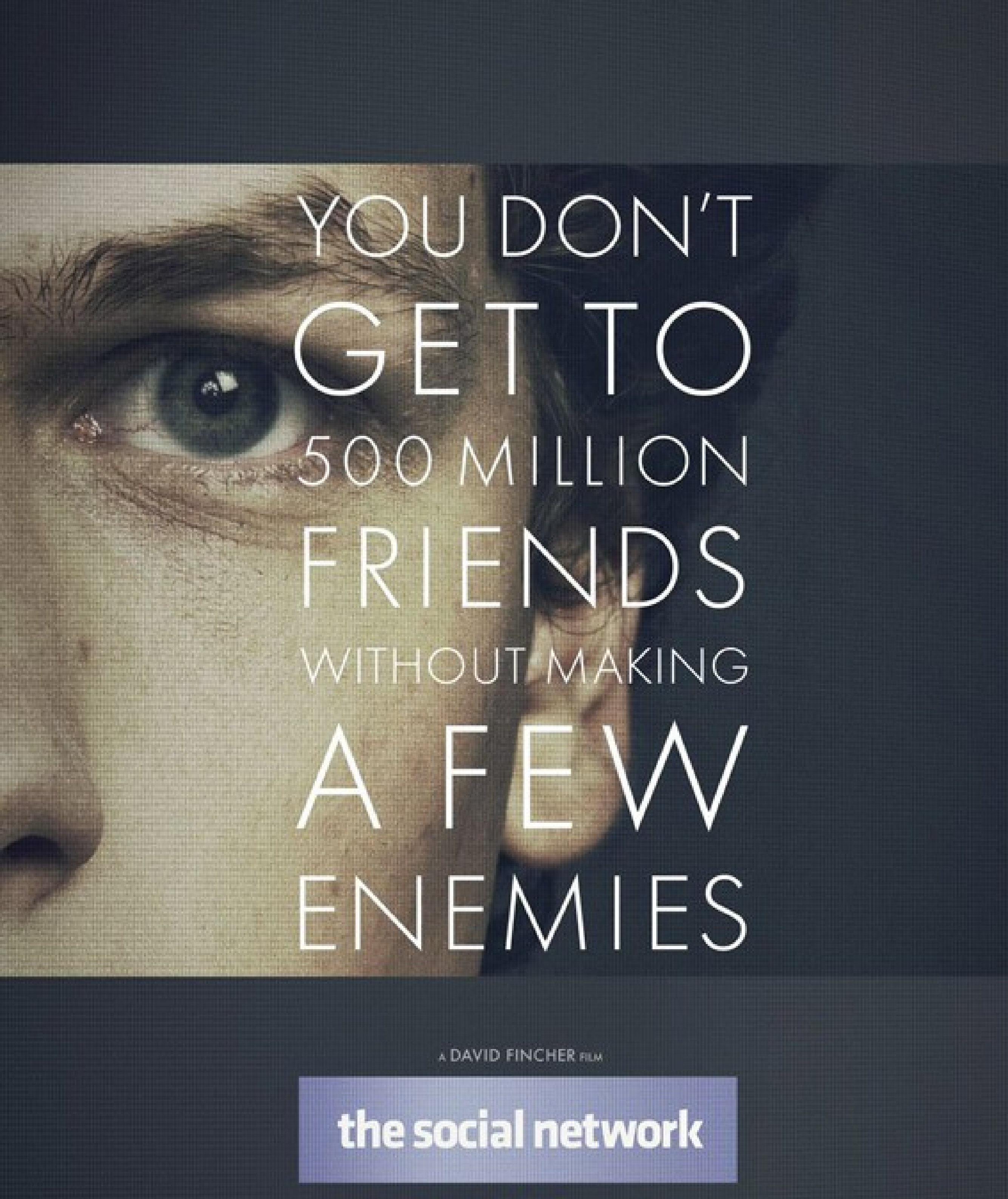
Graphs

Friday Four Square!
4:15PM, Outside Gates

Announcements

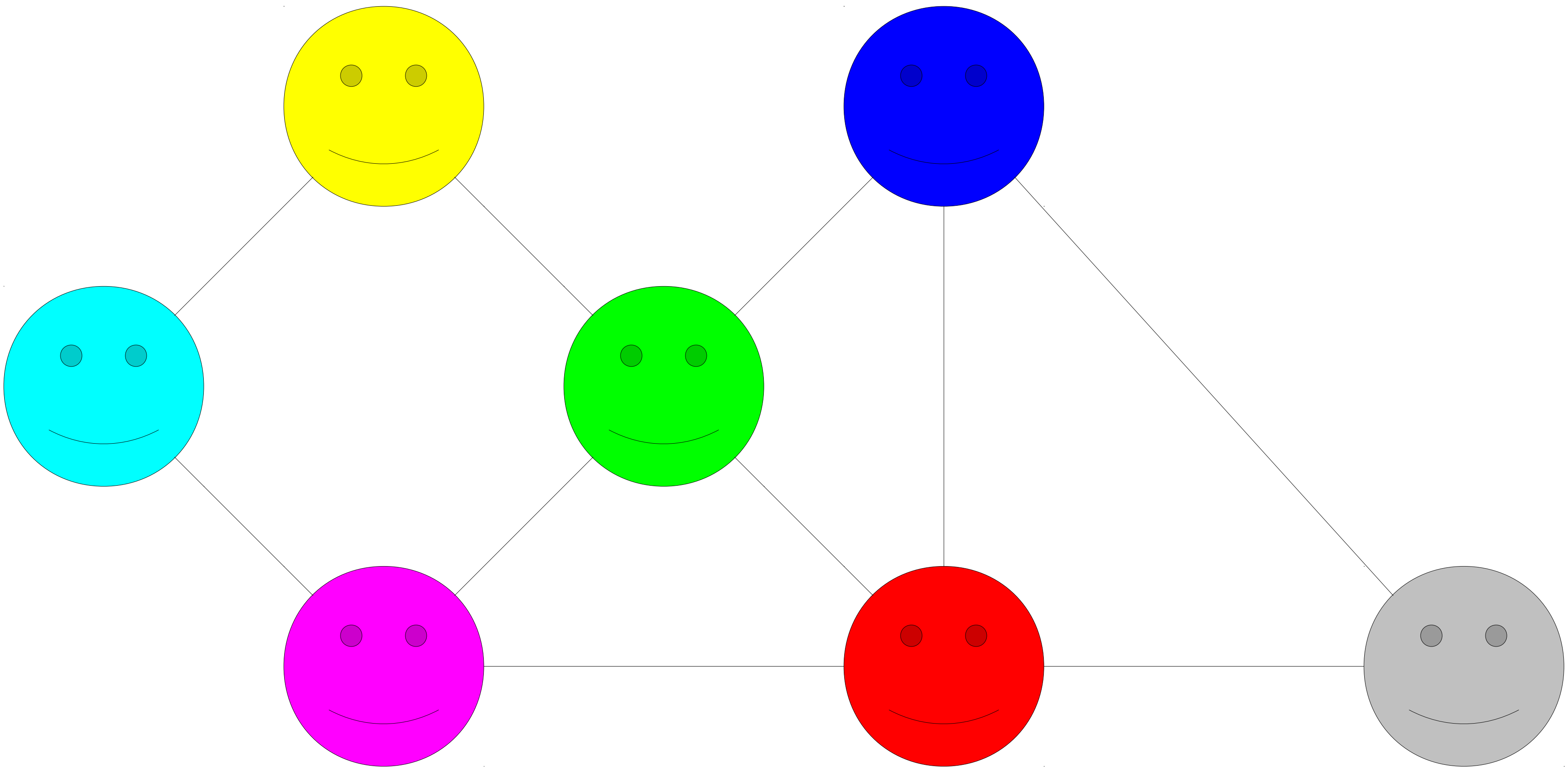
- Second midterm is next **Thursday, May 31**.
- Exam location by last name:
  - A – F: Go to Hewlett 201.
  - G – Z: Go to Hewlett 200.
- Covers material up through and including today's lecture.
- Comprehensive, but primarily focuses on algorithmic efficiency and data structures.
- Practice exam posted to course website.
- Review session next Tuesday from 7-9PM in Hewlett 201.

In the news...

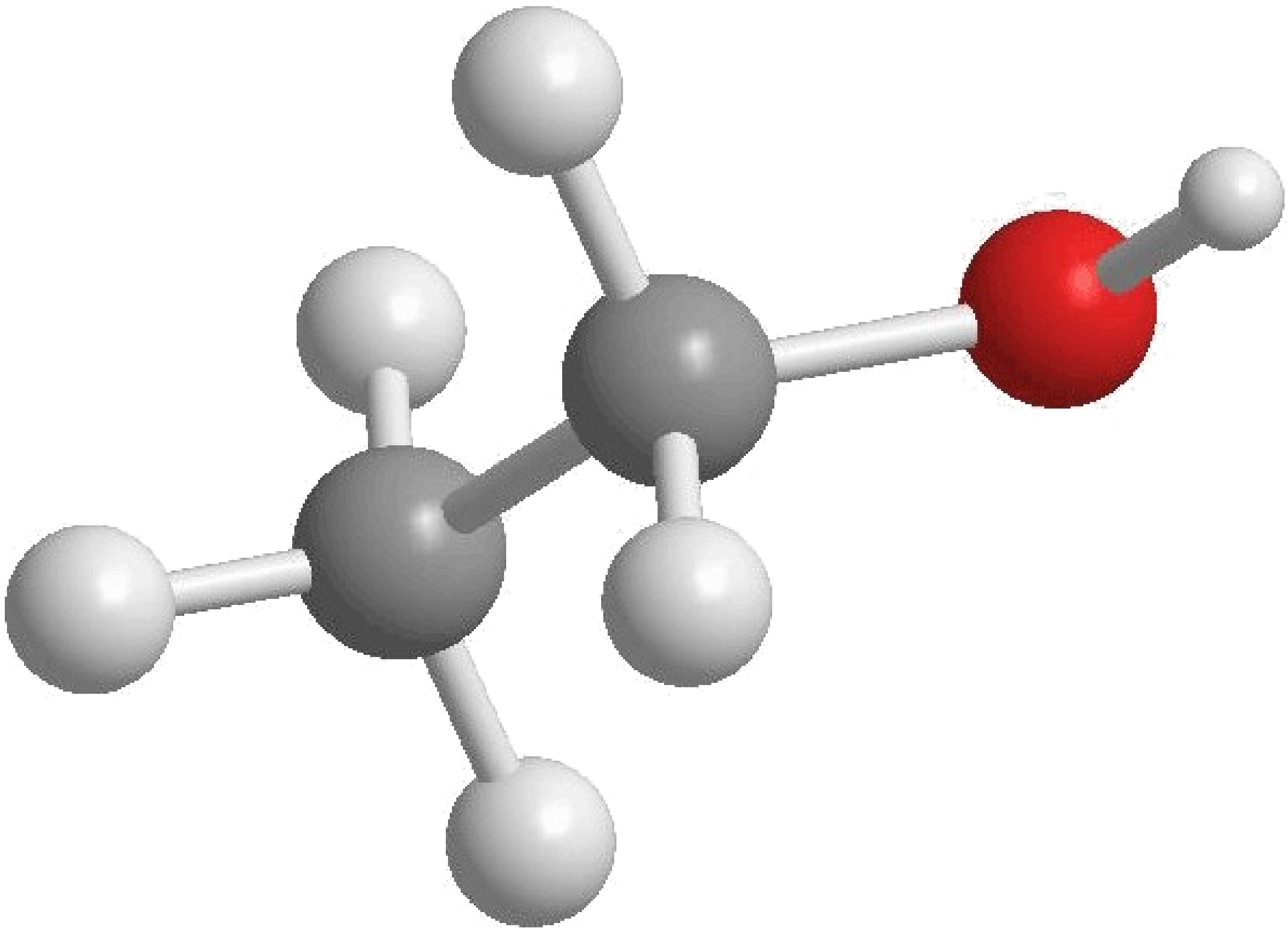

YOU DON'T
GET TO
500 MILLION
FRIENDS
WITHOUT MAKING
A FEW
ENEMIES
A DAVID FINCHER FILM
the social network


facebook

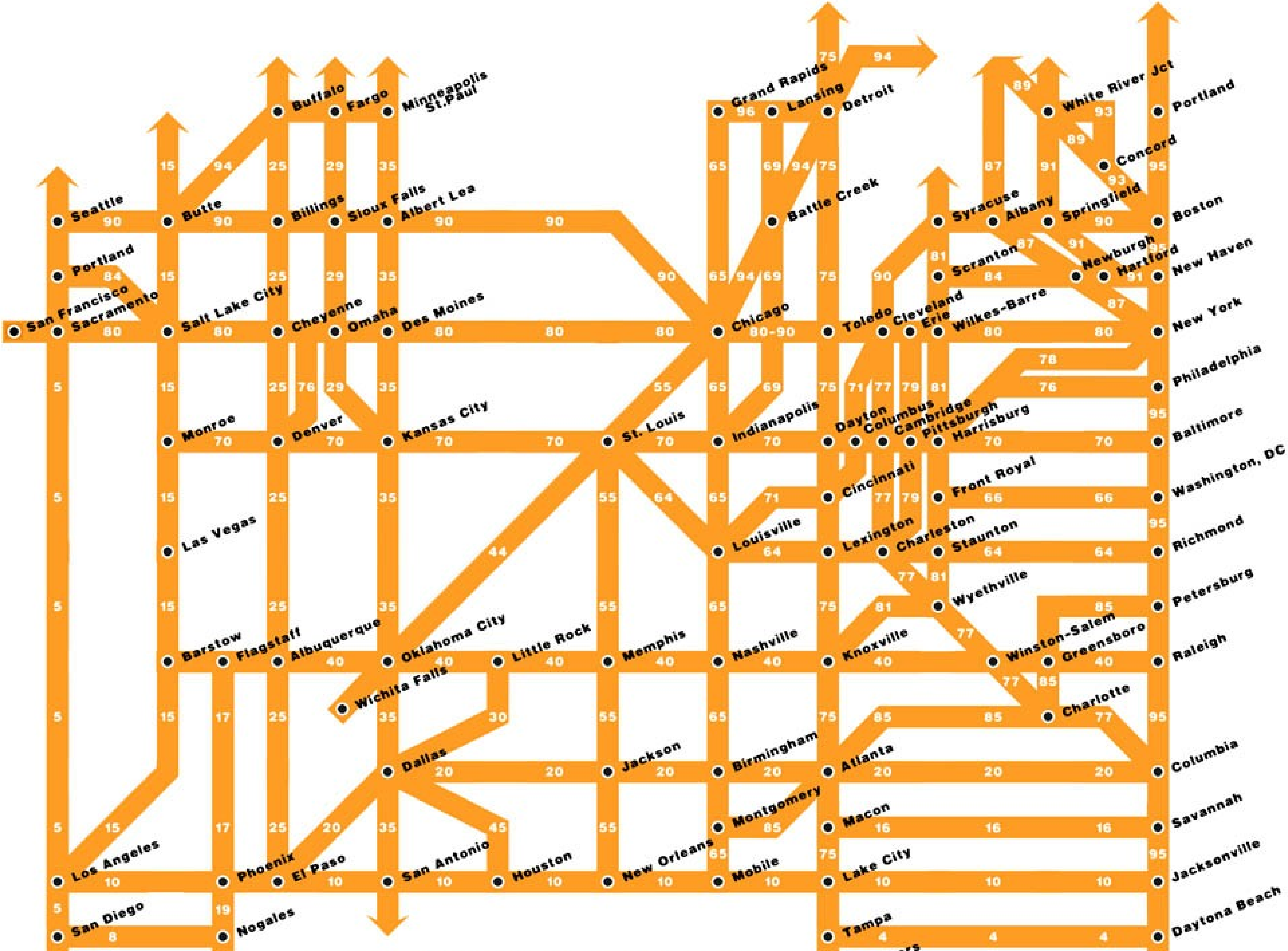
A Social Network



Chemical Bonds



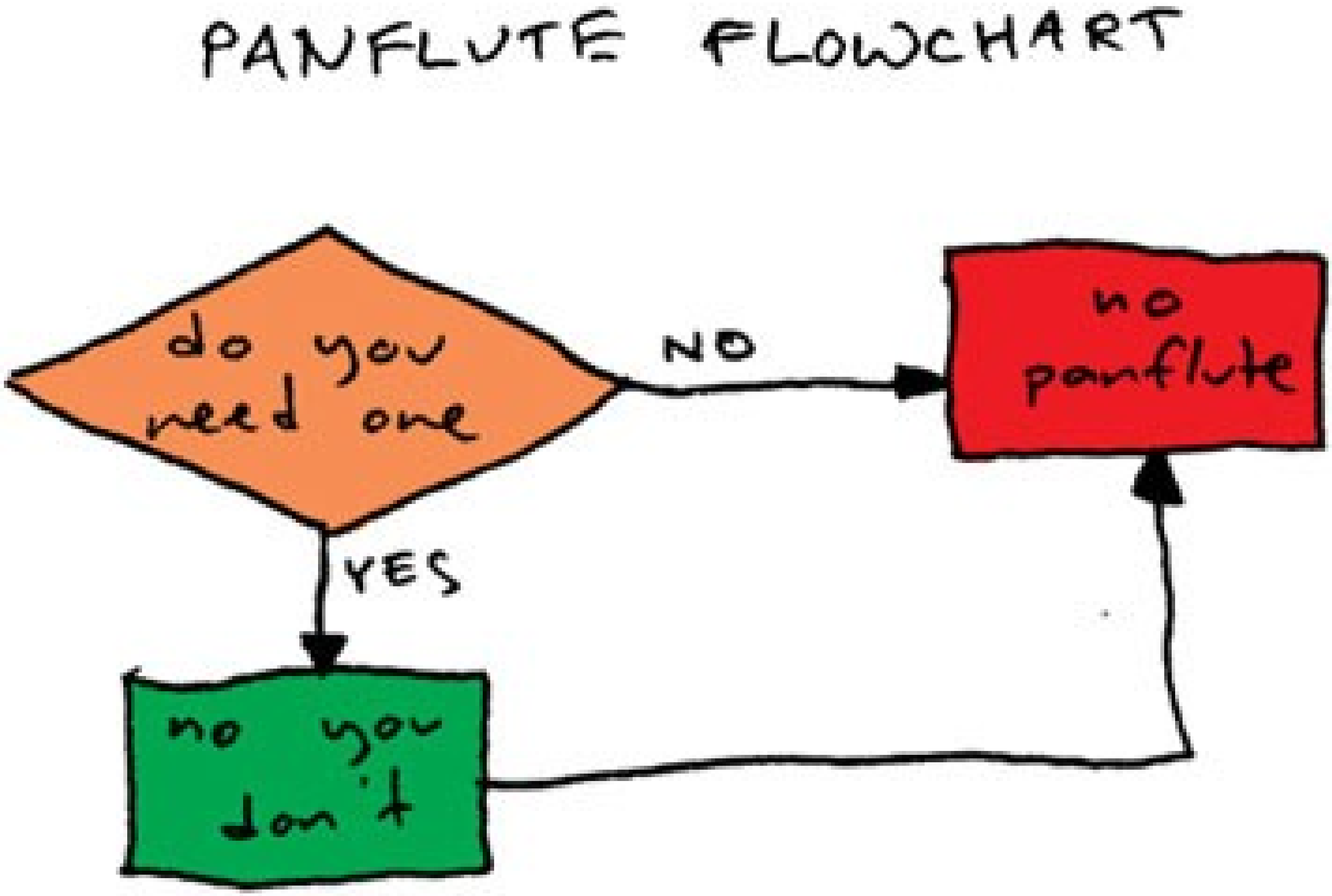
http://4.bp.blogspot.com/-xCtBJ8lKHqA/Tjm0BONWBRI/AAAAAAAAAK4/-mHrbAUOHHg/s1600/Ethanol2.gif



THE EISENHOWER INTERSTATE SYSTEM

(simplified)

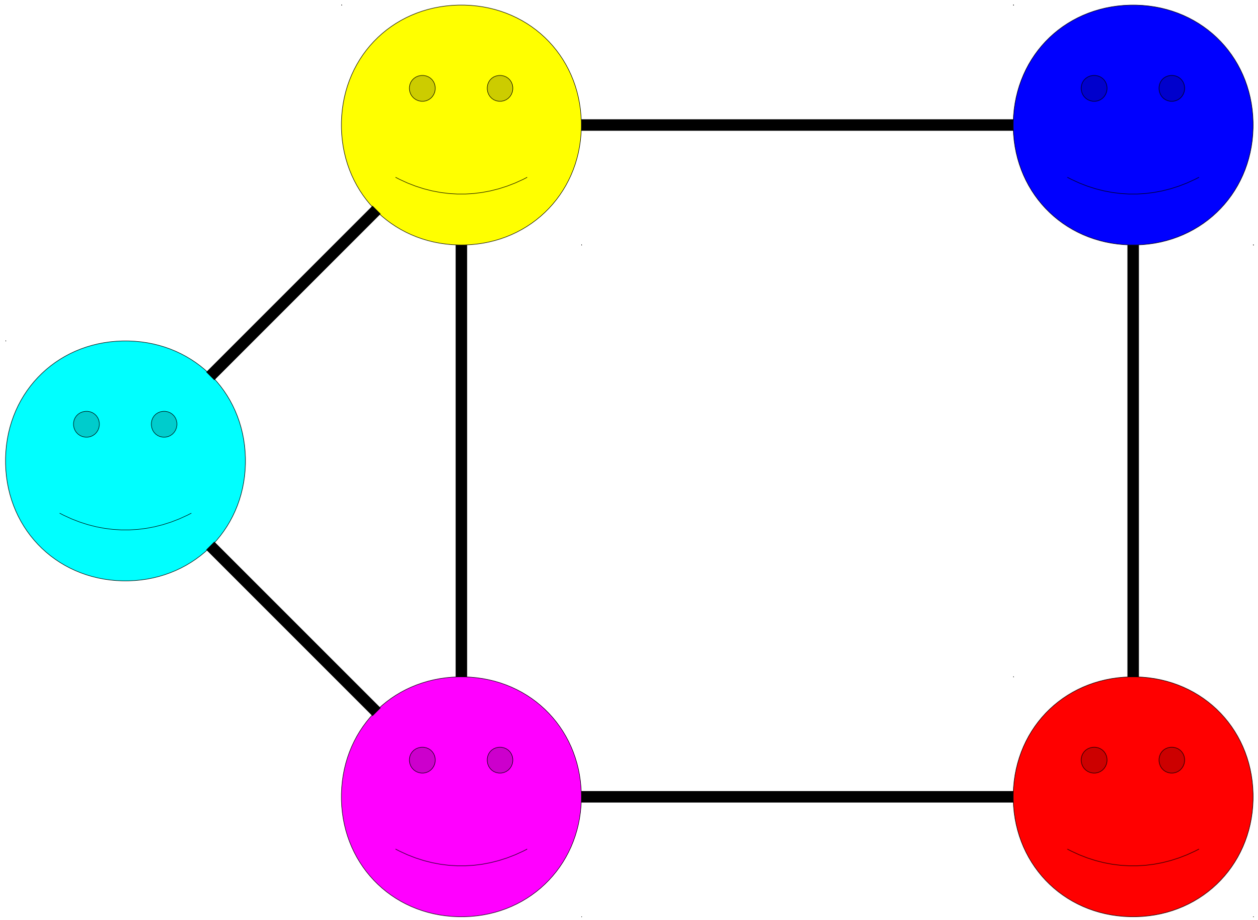
CHRIS YATES 2007


PANFLUTE FLOWCHART
do you need one
NO
no panflute
YES
no you don't

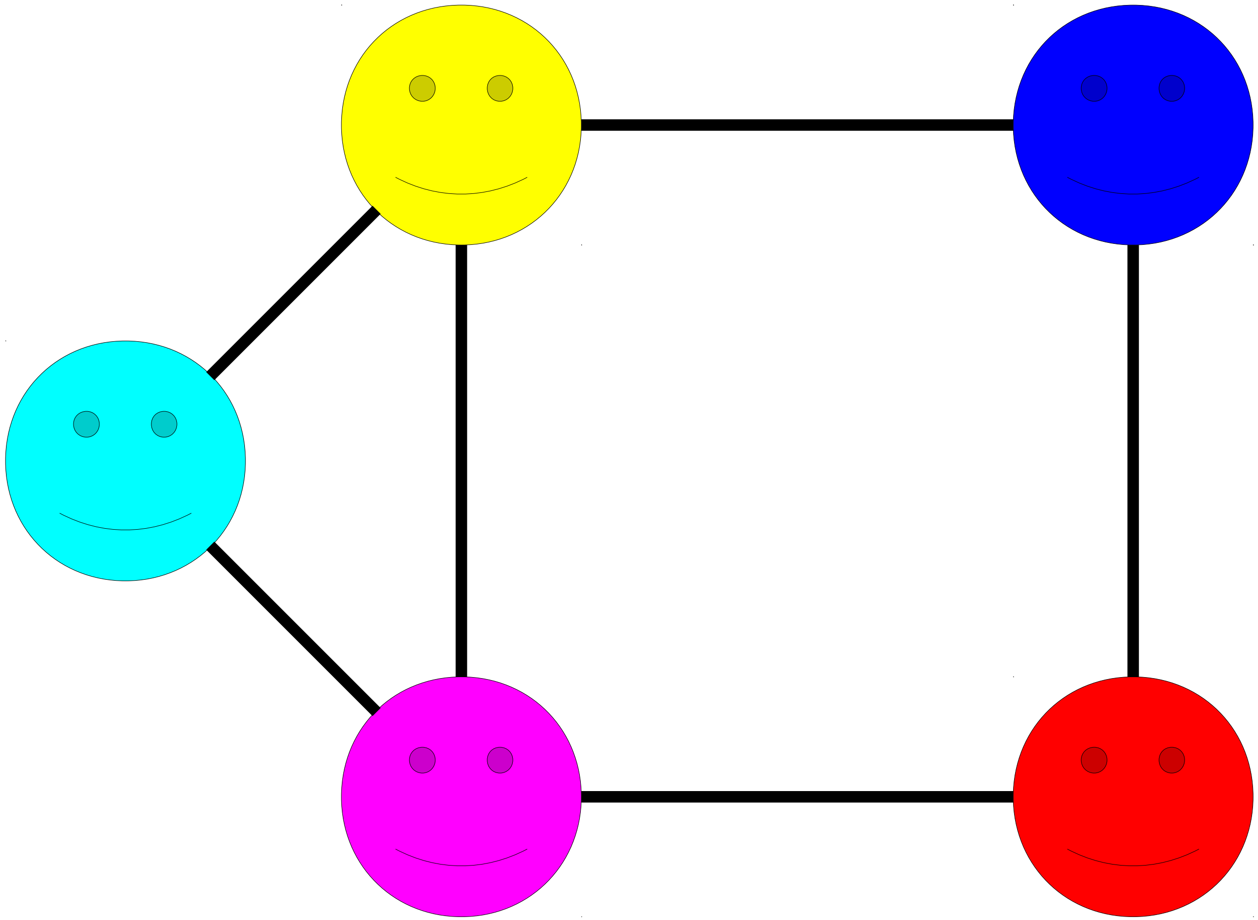
A **graph** is a mathematical structure for representing relationships.

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

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

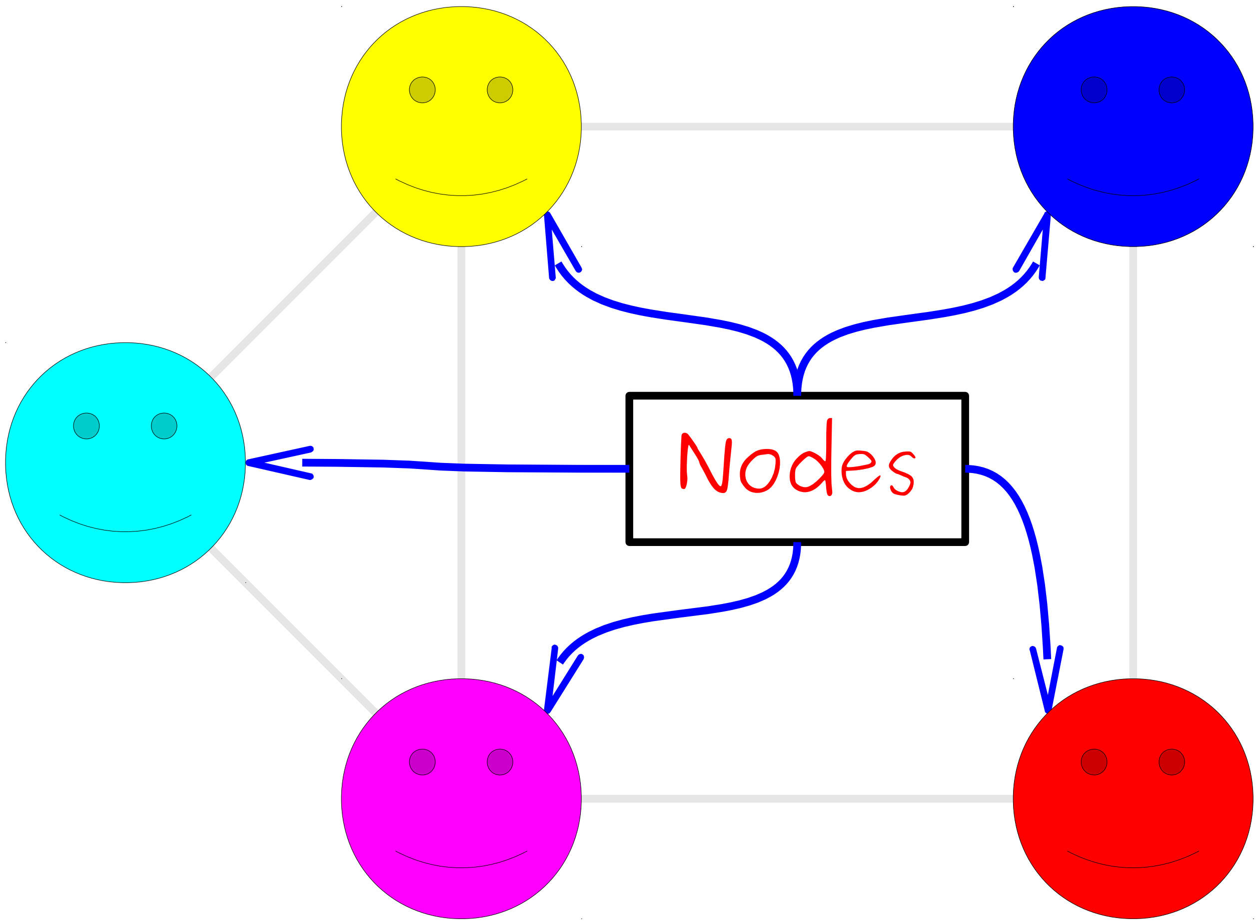


A **graph** is a mathematical structure for representing relationships.



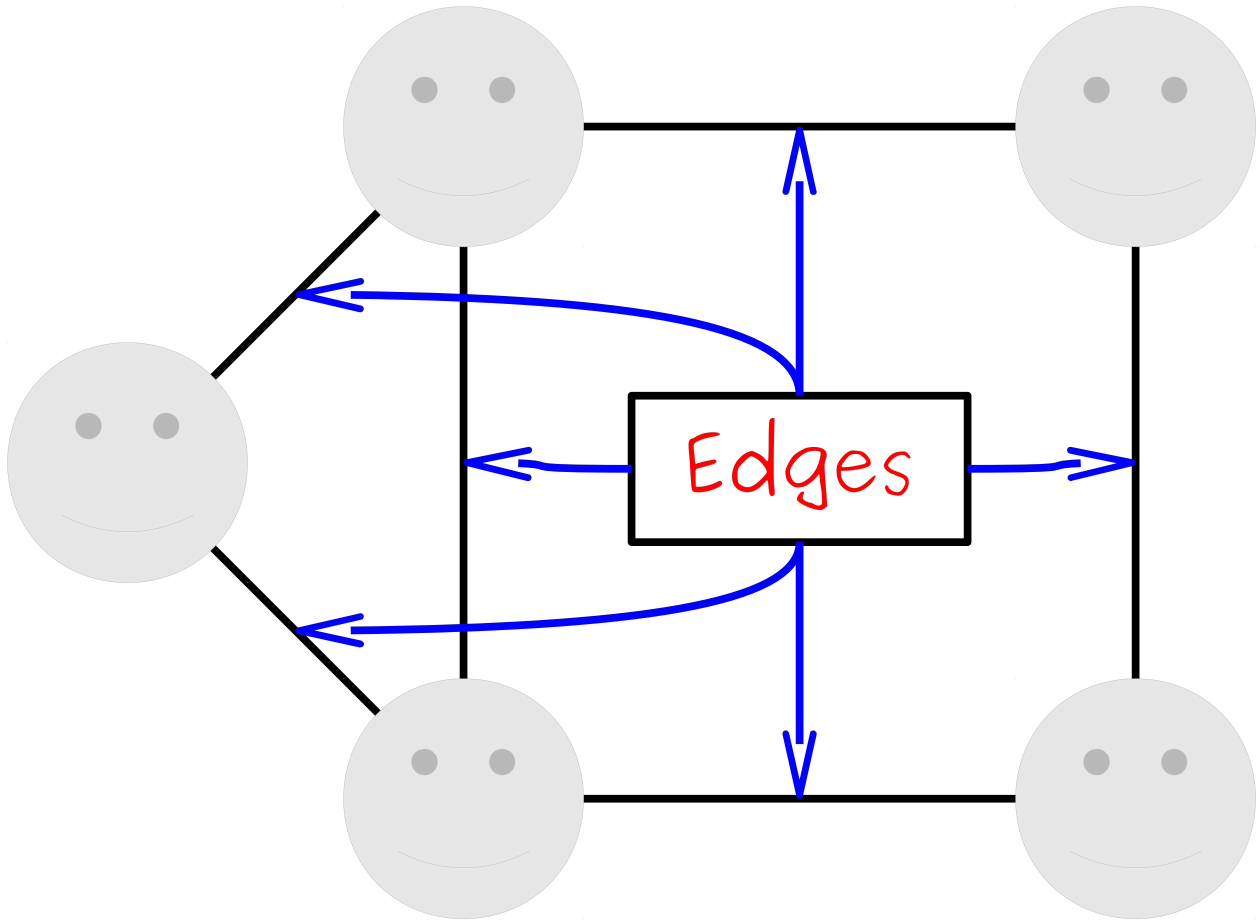
A graph consists of a set of **nodes** connected by **edges**.

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



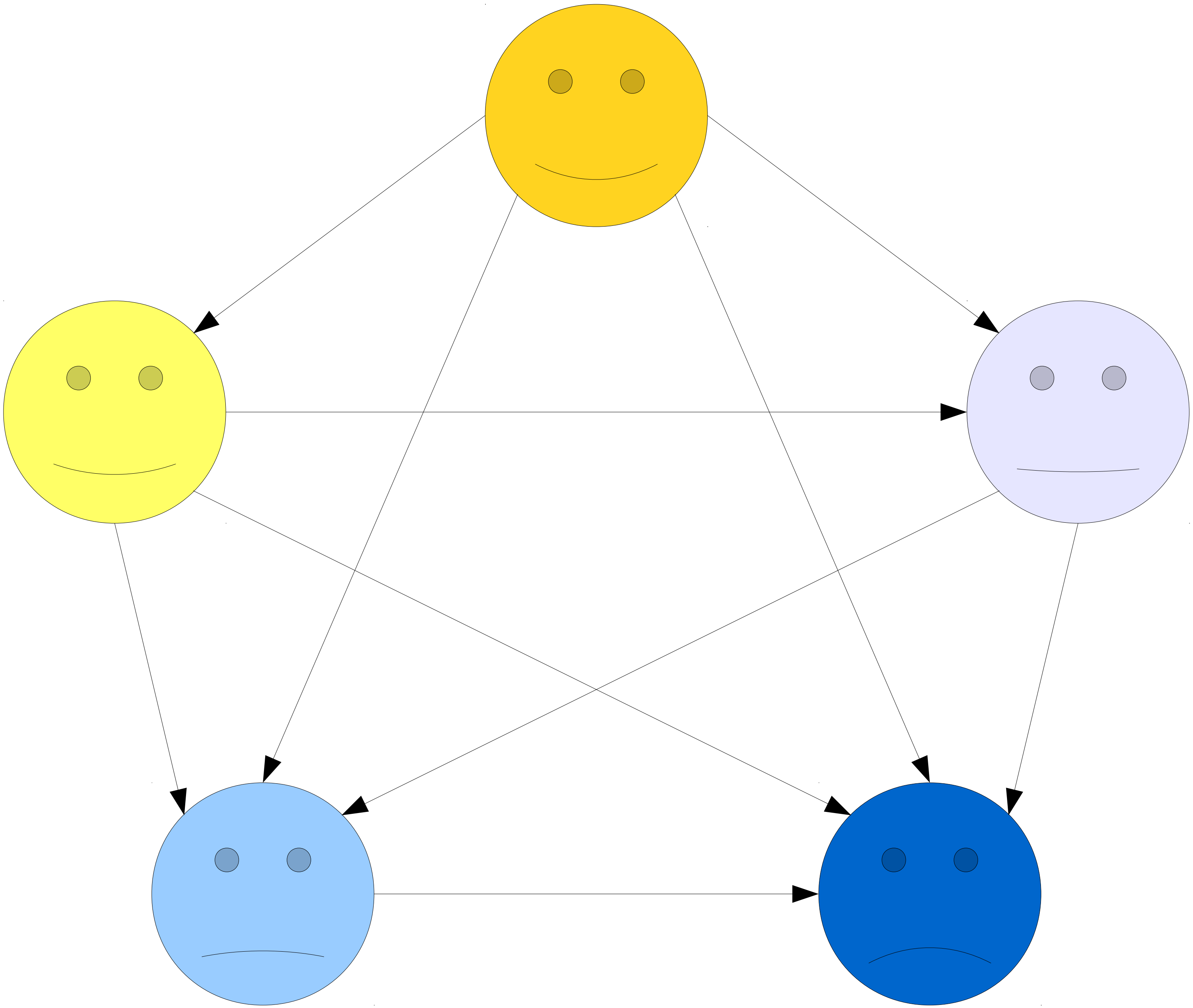
A graph consists of a set of **nodes** connected by **edges**.

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

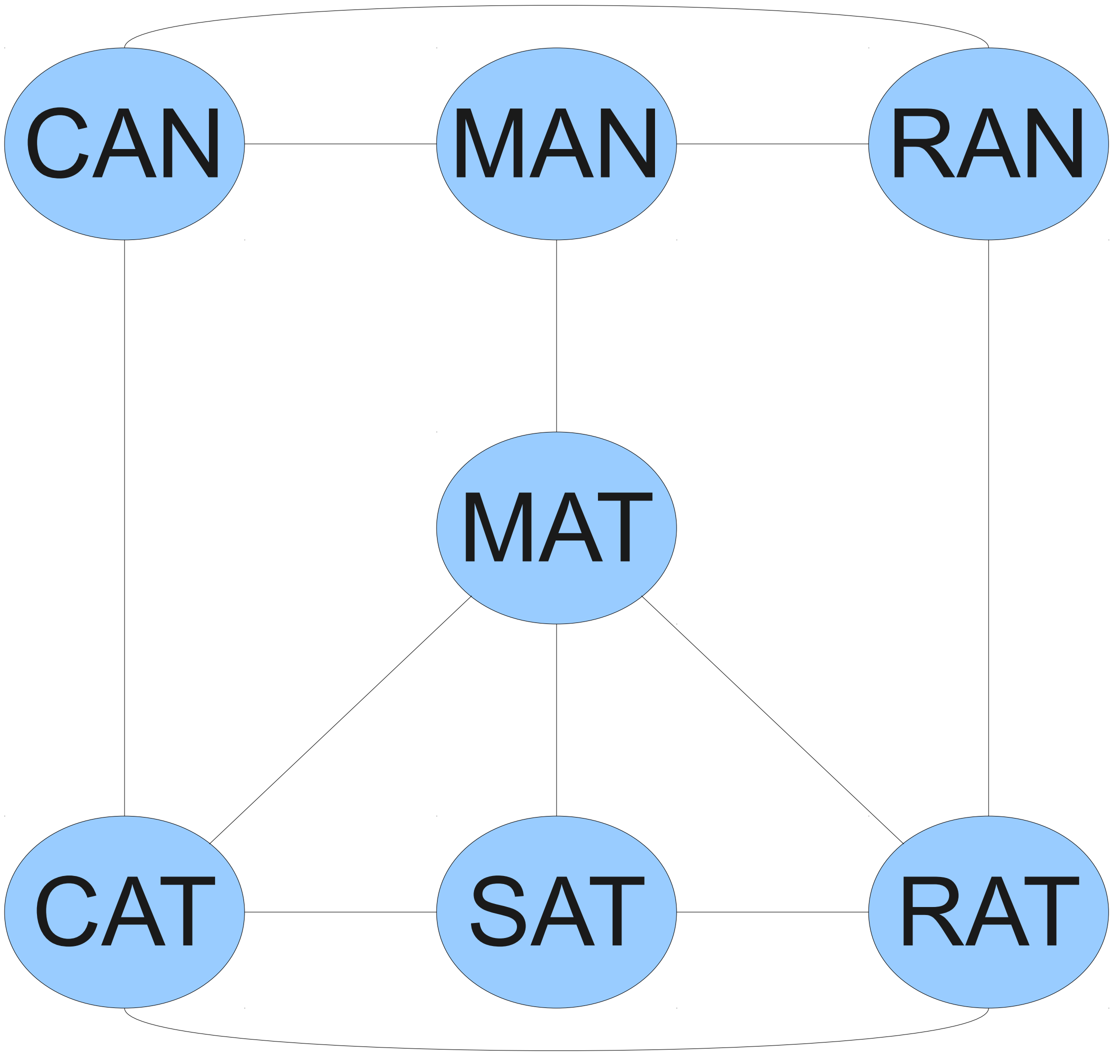


A graph consists of a set of **nodes** connected by **edges**.

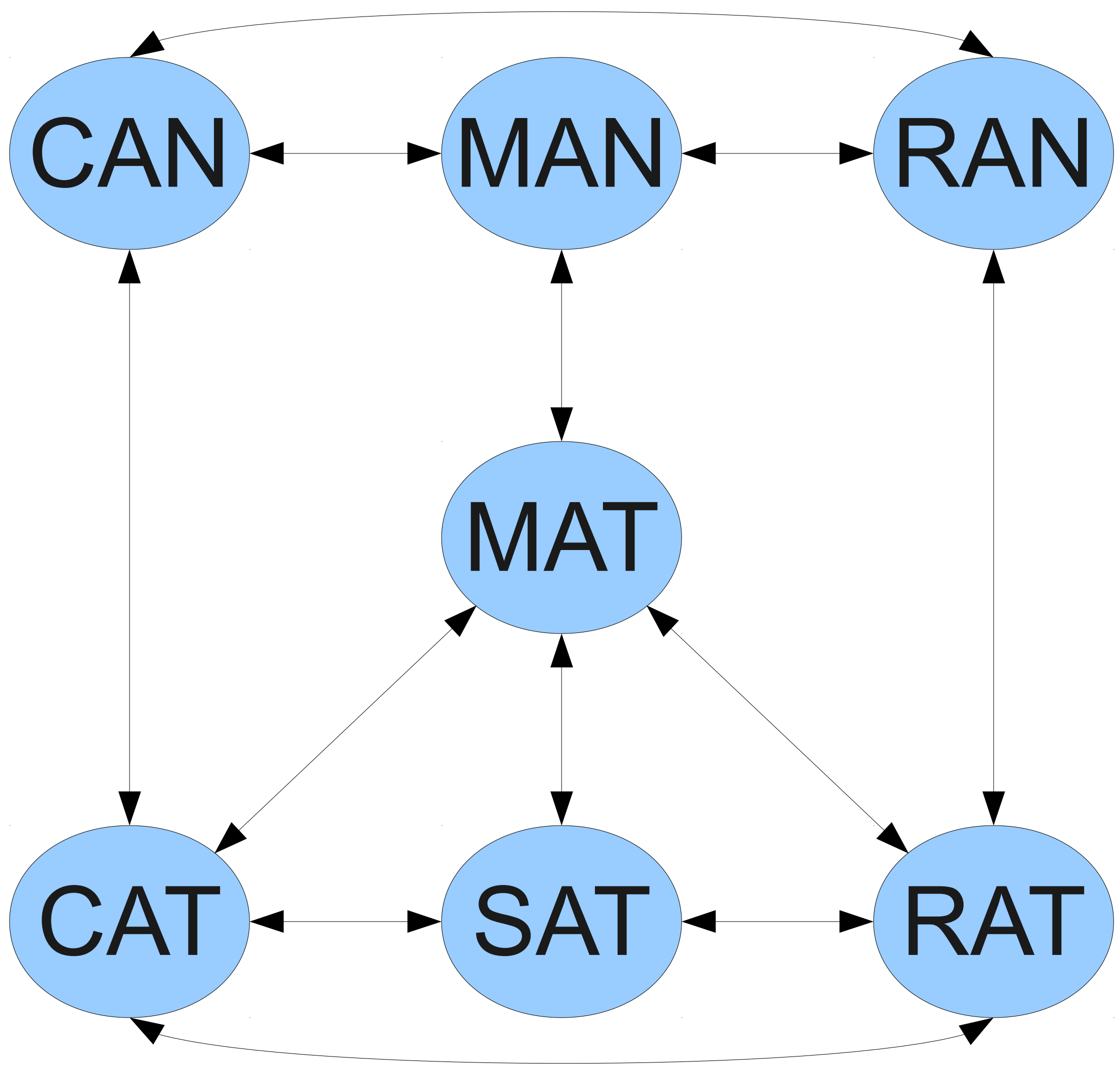
Some graphs are **directed**.



Some graphs are **undirected**.



Some graphs are **undirected**.

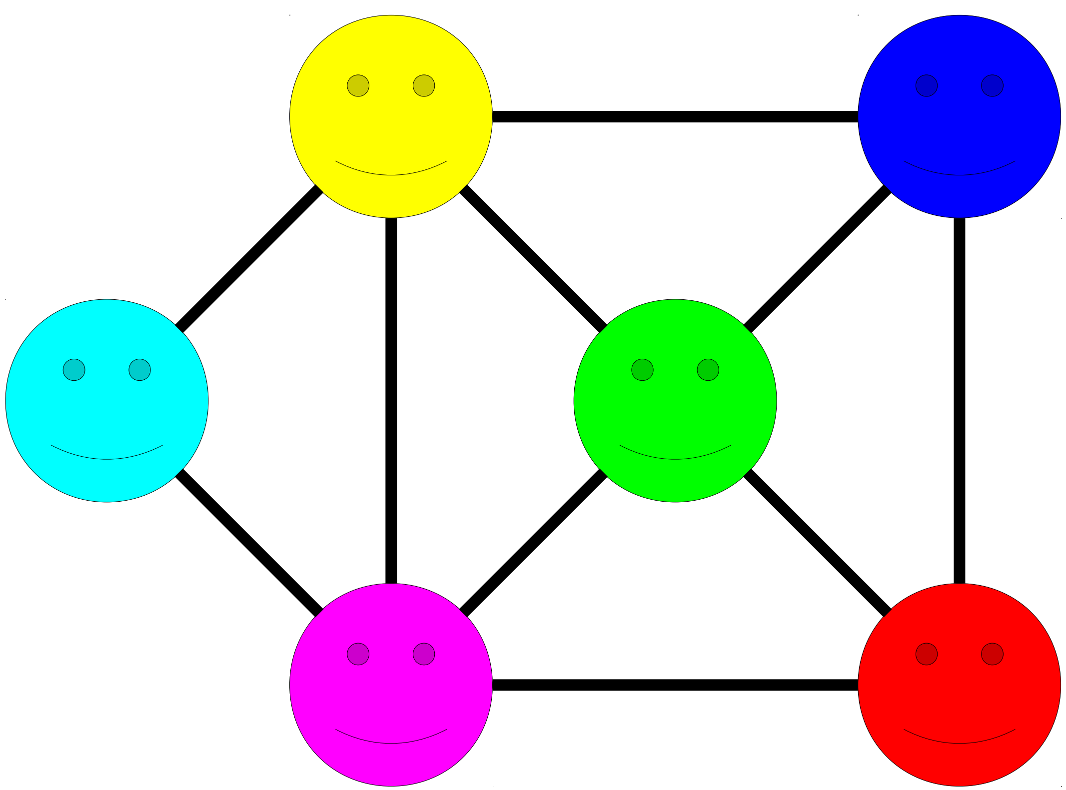


You can think of them as directed graphs with edges both ways.

How can we represent graphs in C++?

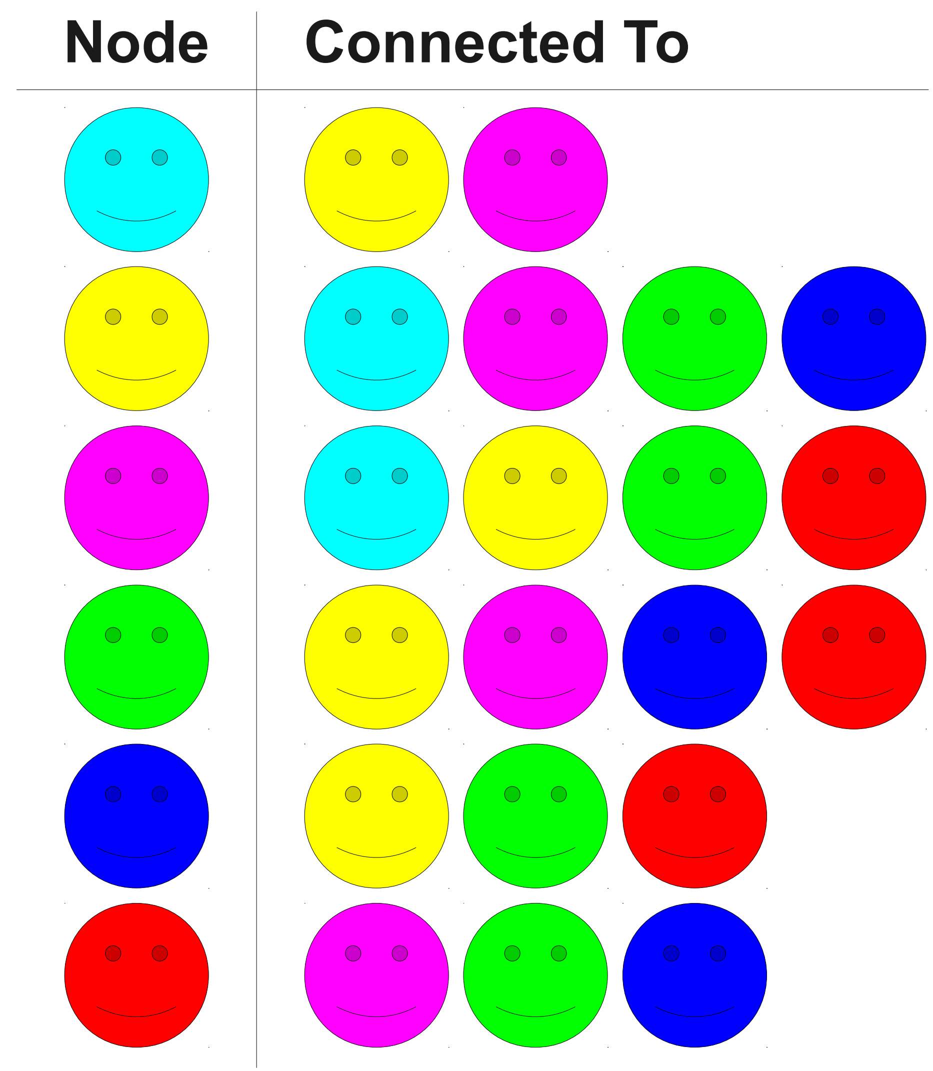
Representing Graphs

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*>
```



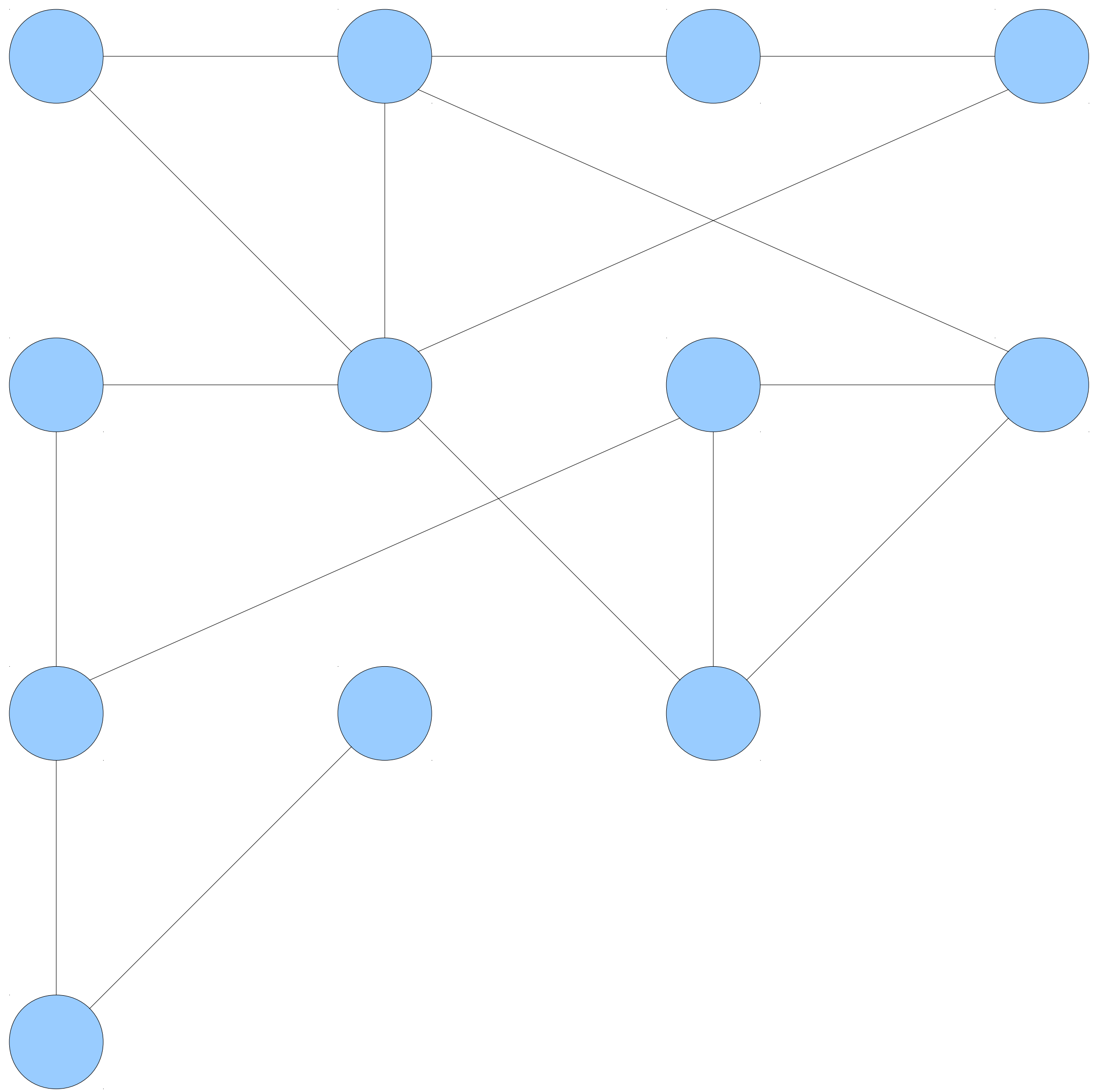
The Wikipedia Graph

- 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.

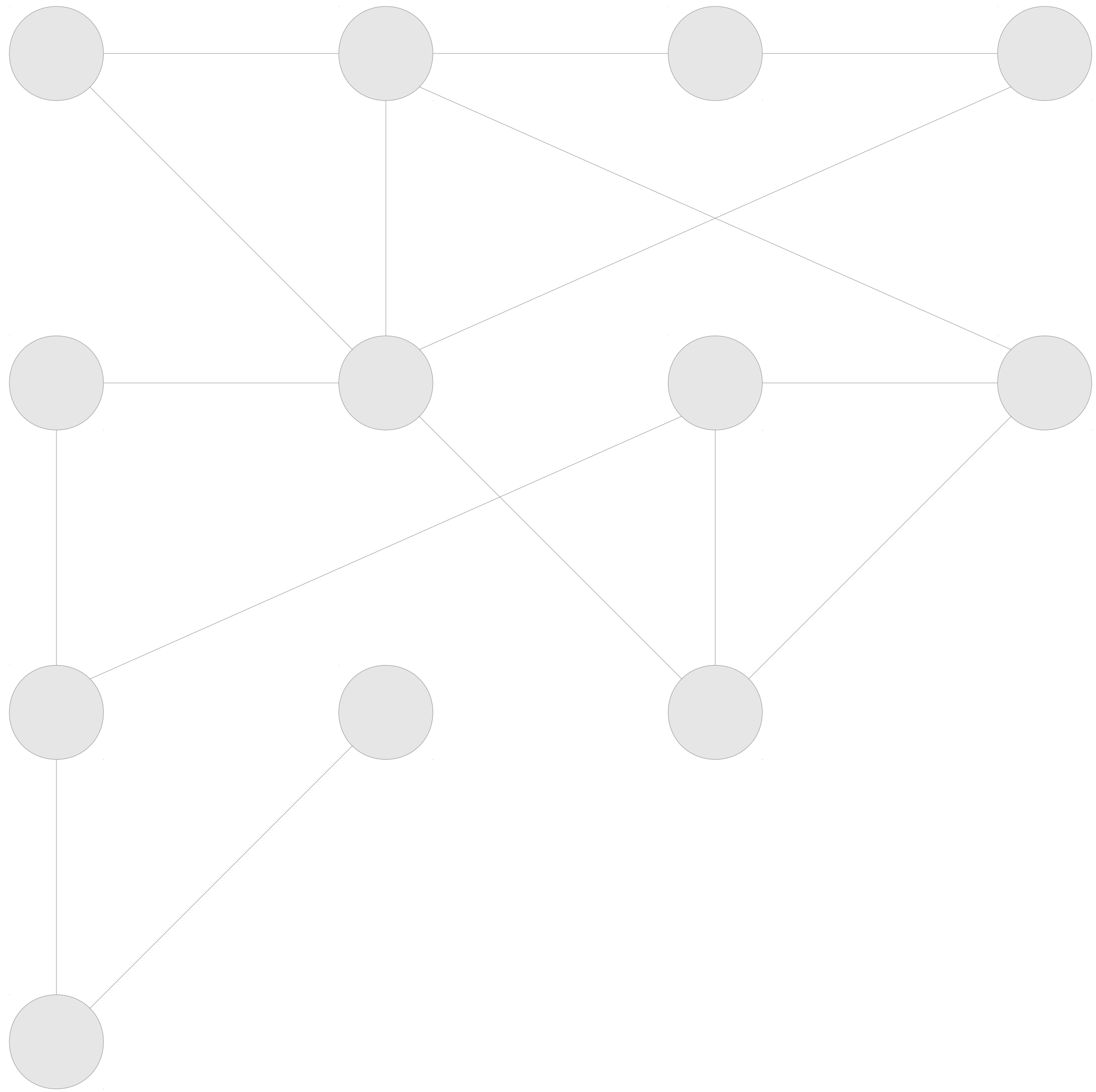
Iterating over a Graph

- Given a linked list, there was just one way to traverse the list.
  - Keep going forward.
- In a binary search tree, we saw three traversals:
  - Preorder
  - Inorder
  - Postorder.
- There are *many* ways to iterate over a graph.

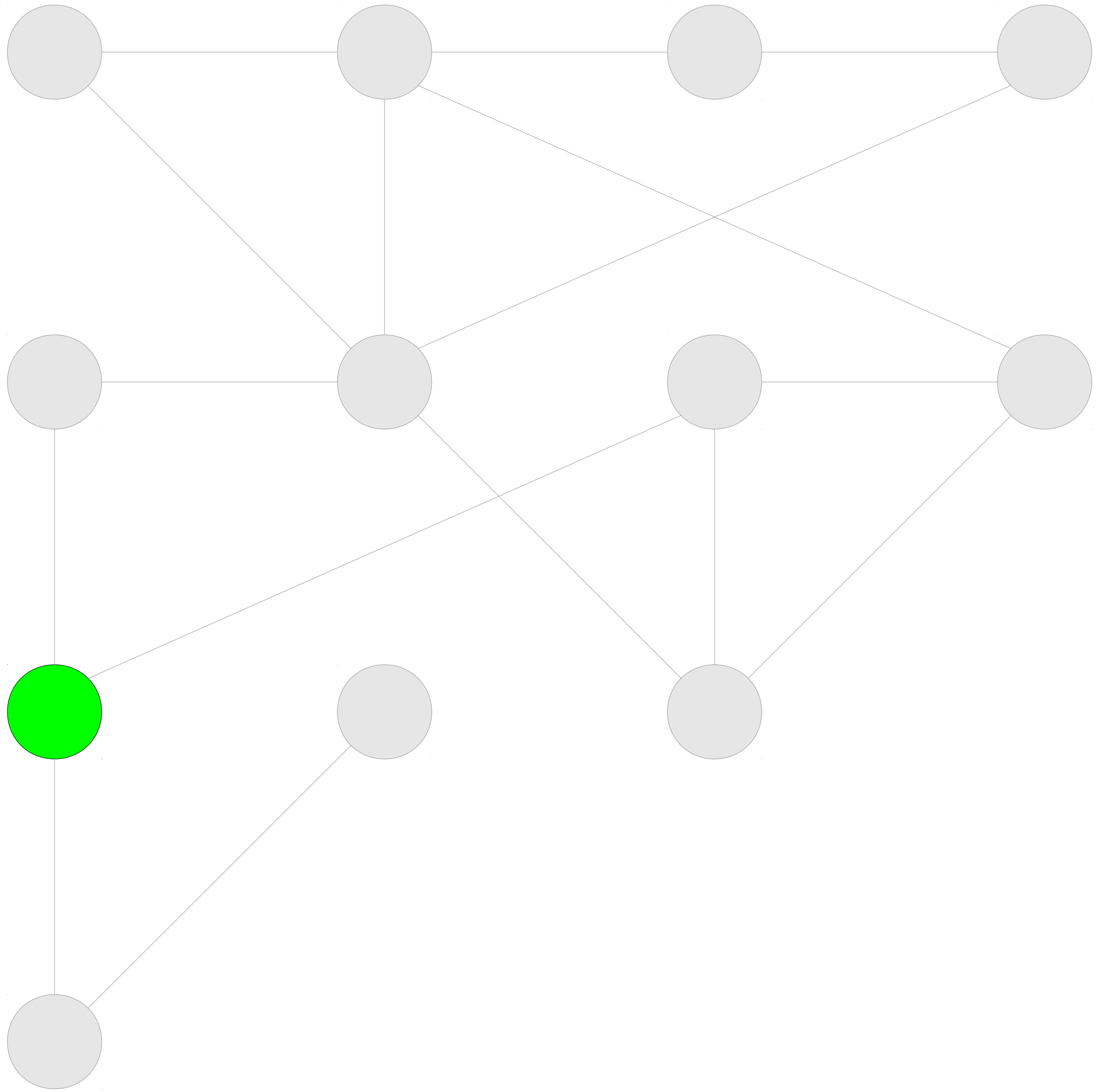
Iterating over a Graph



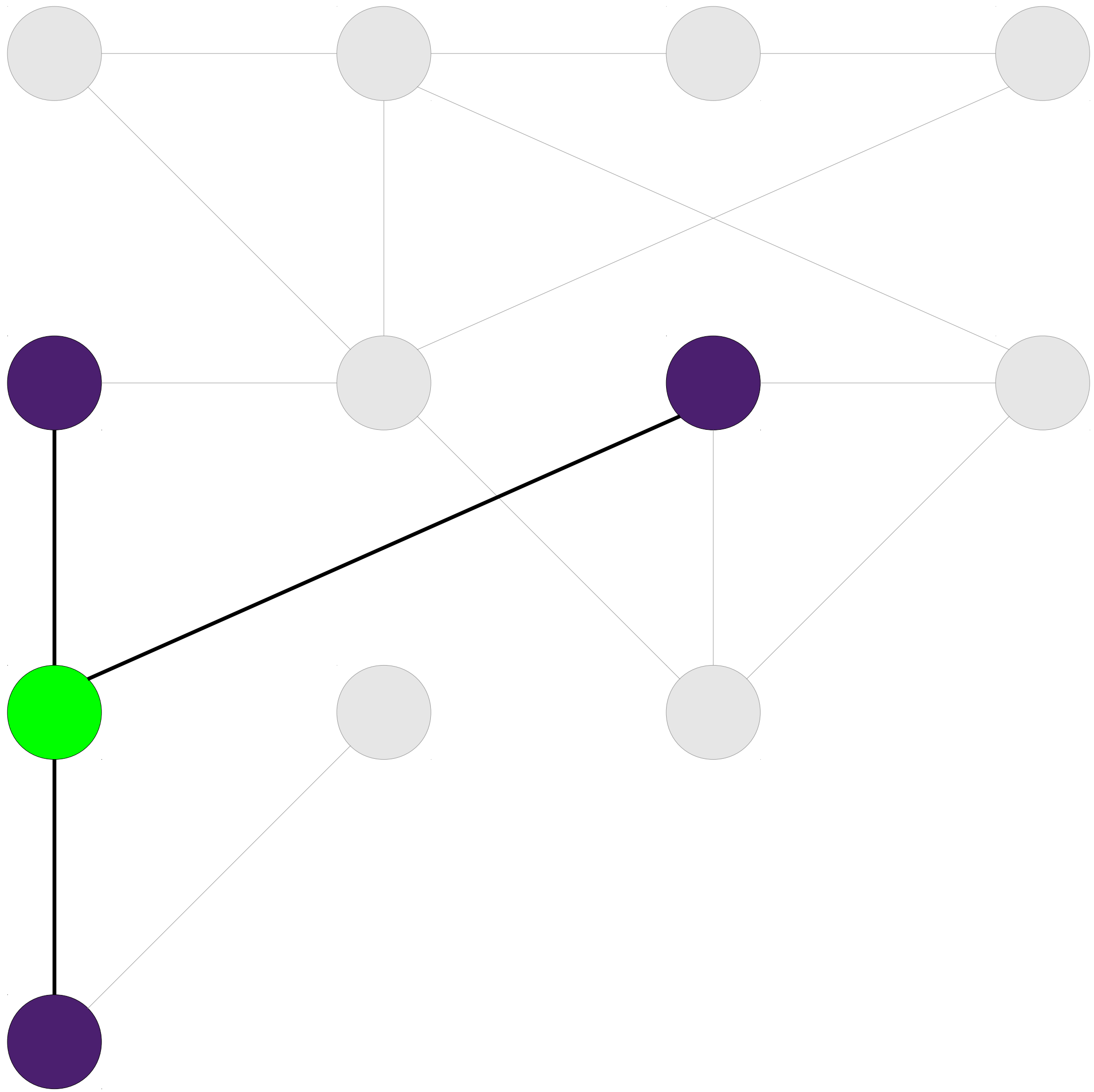
Iterating over a Graph



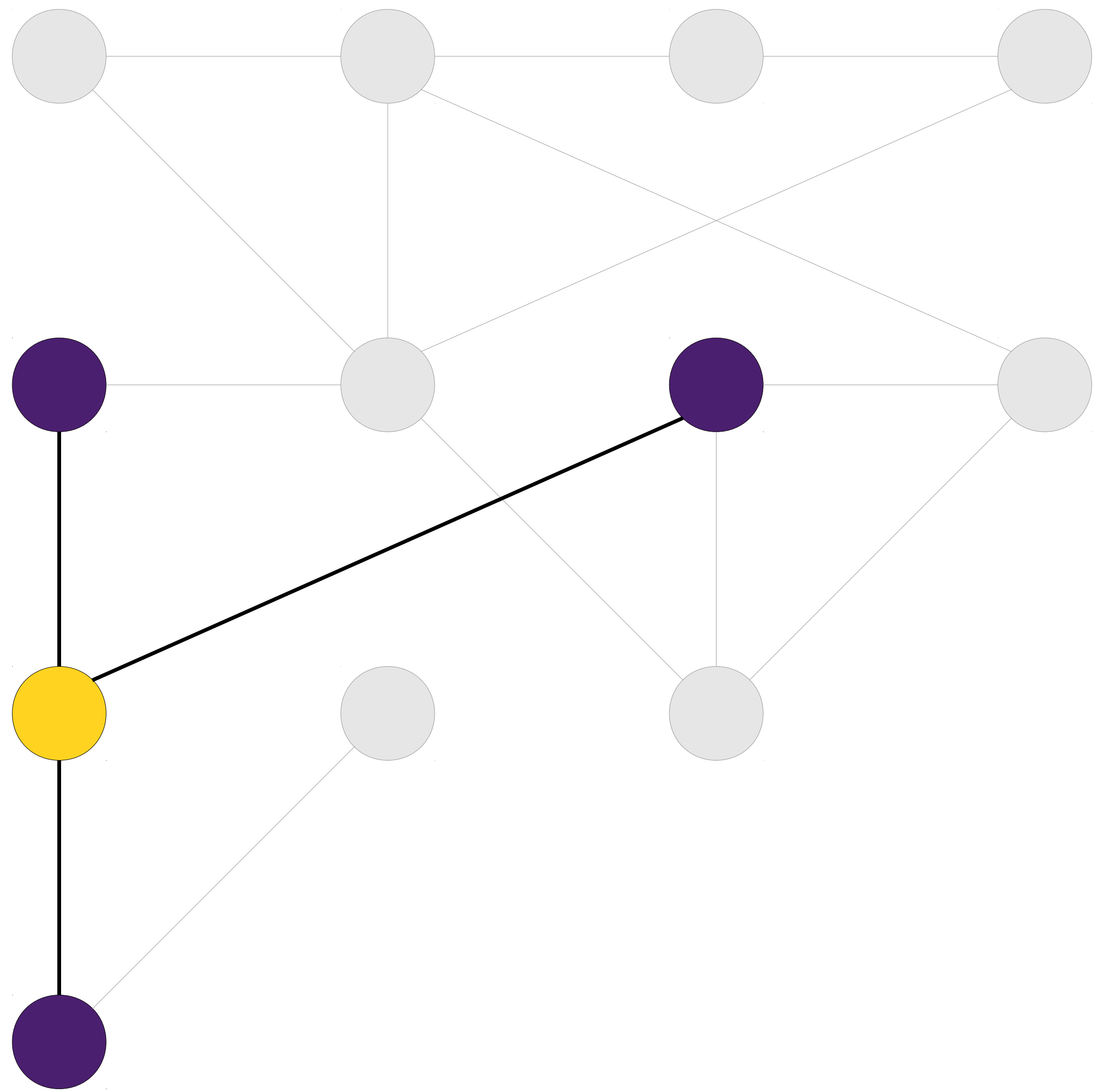
Iterating over a Graph



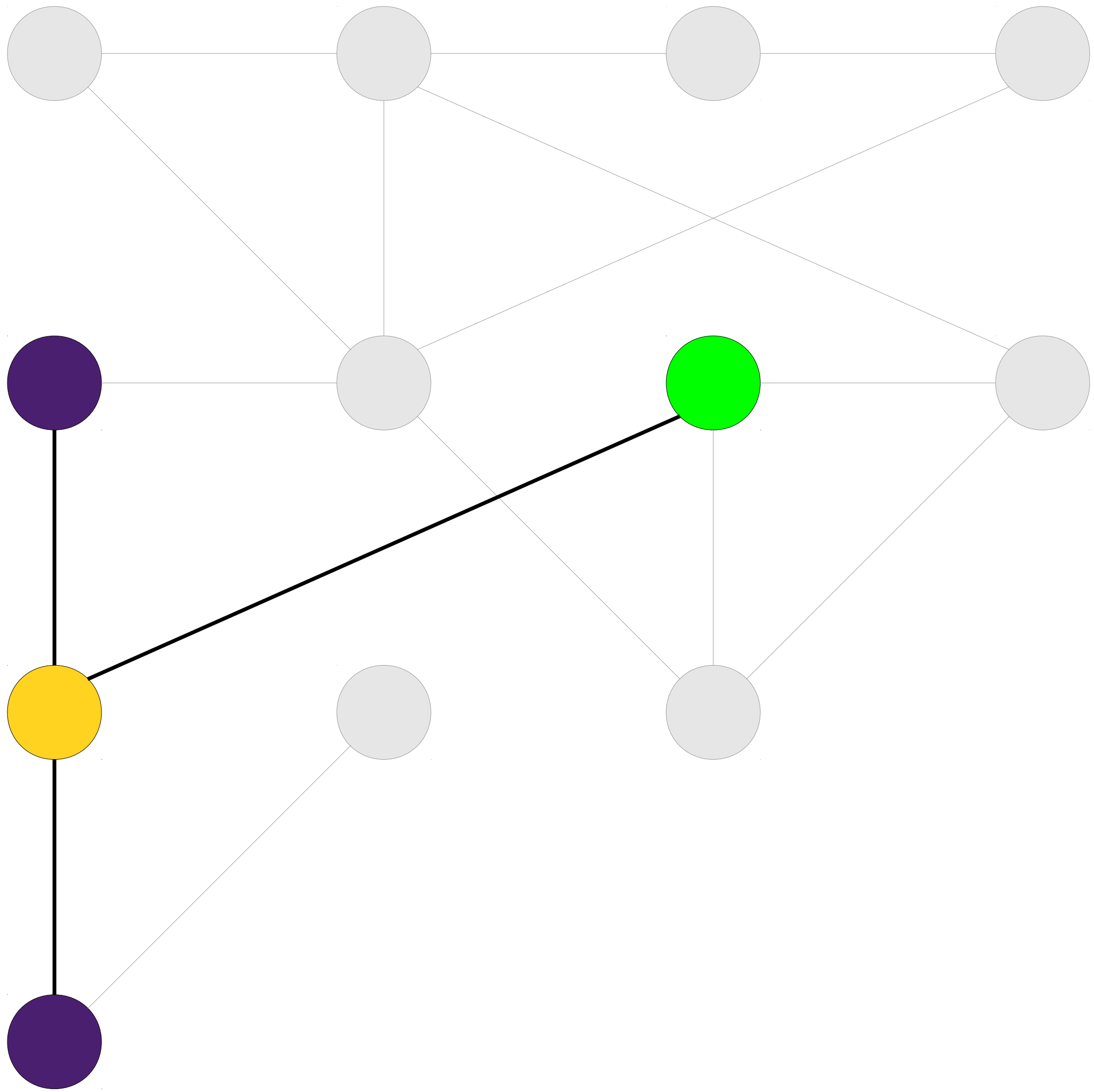
Iterating over a Graph



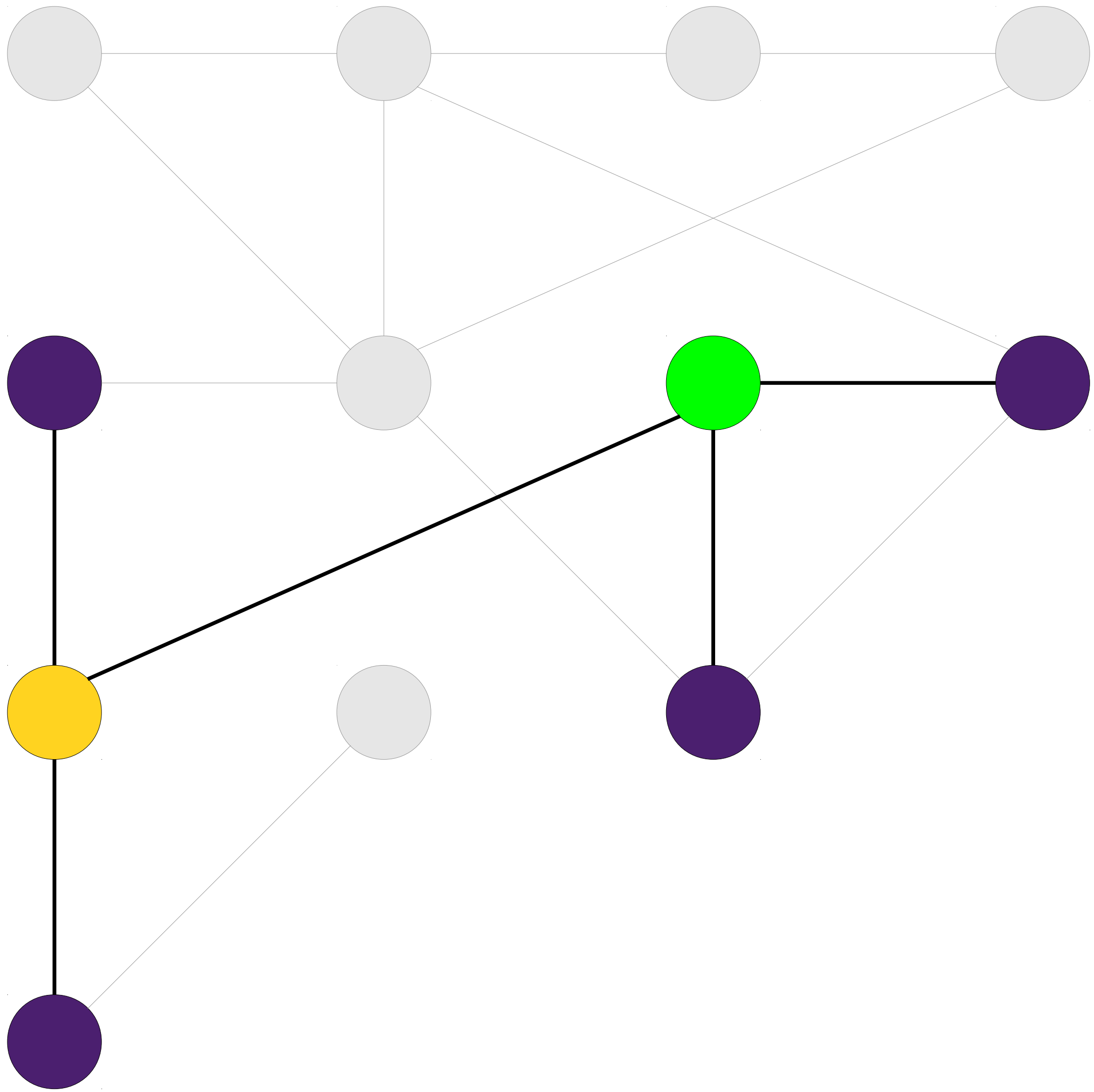
Iterating over a Graph



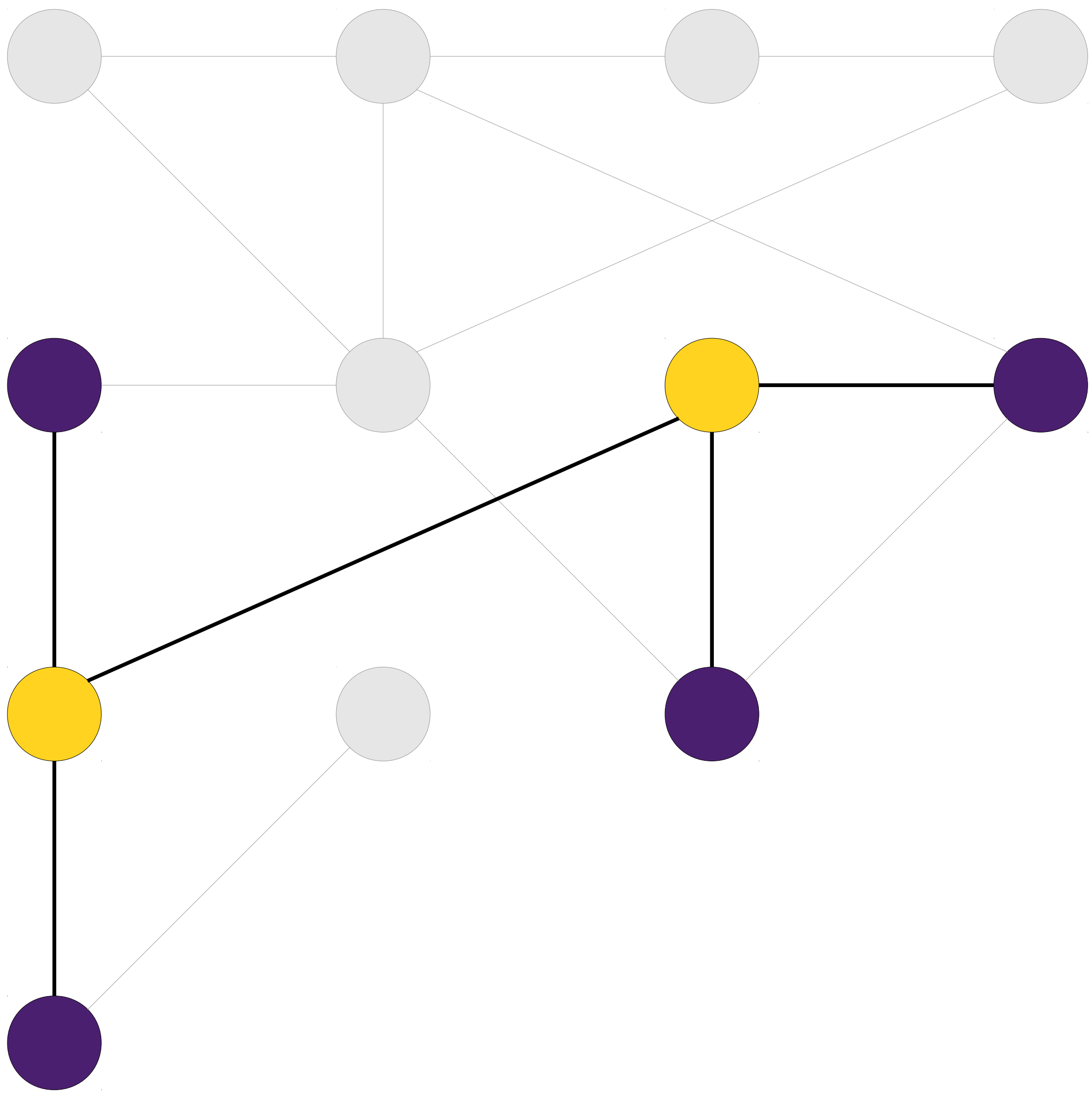
Iterating over a Graph



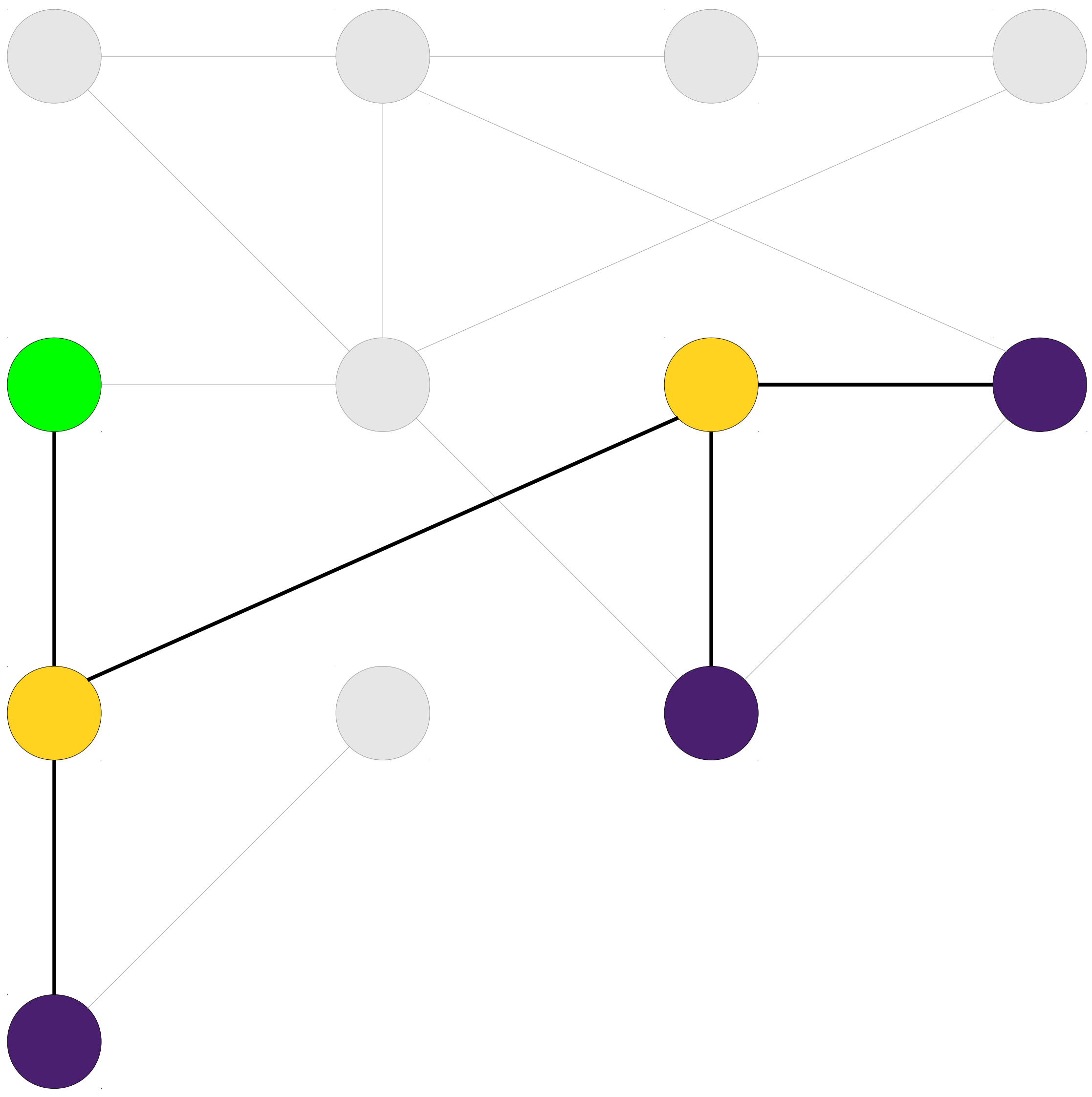
Iterating over a Graph



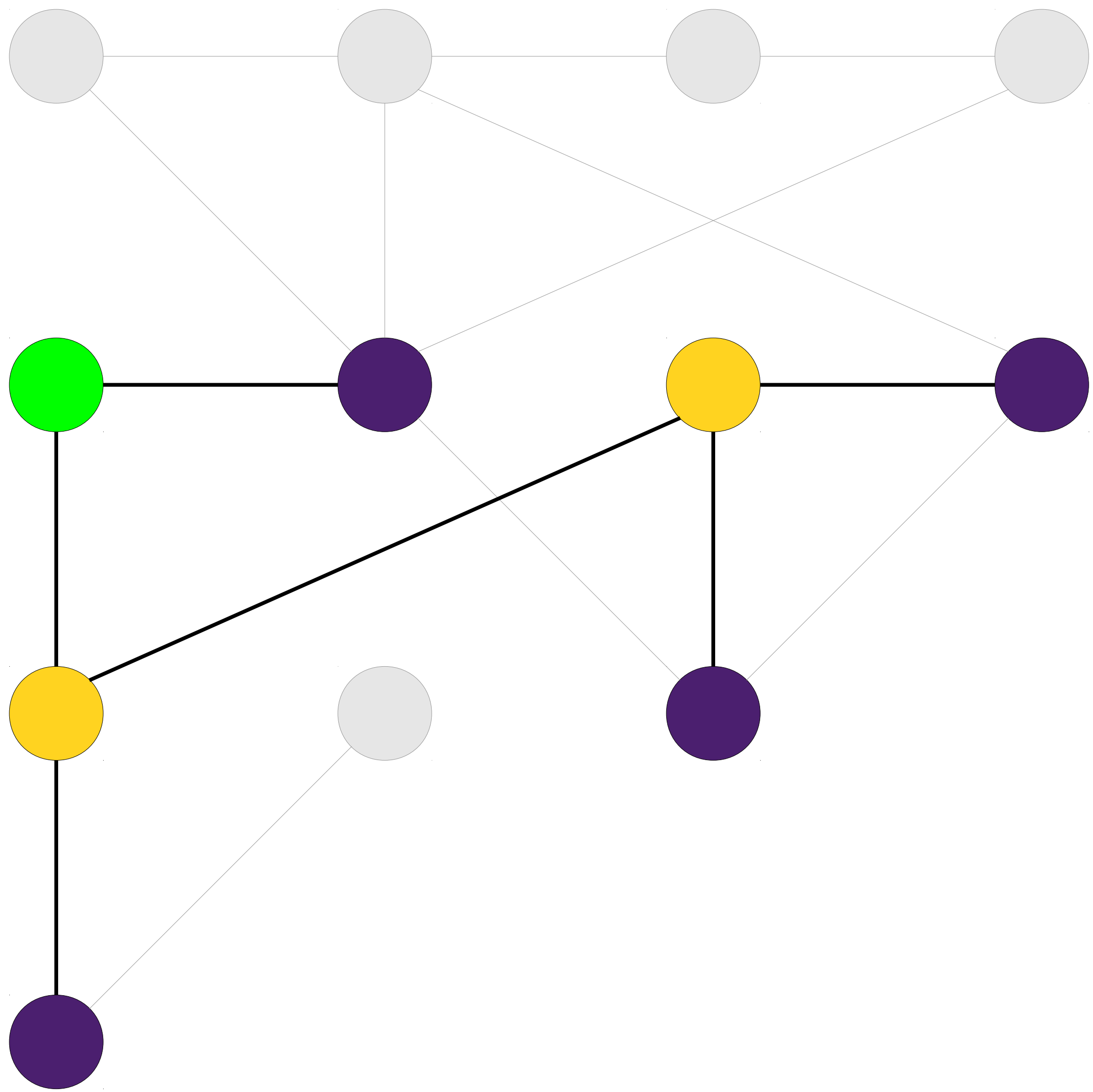
Iterating over a Graph



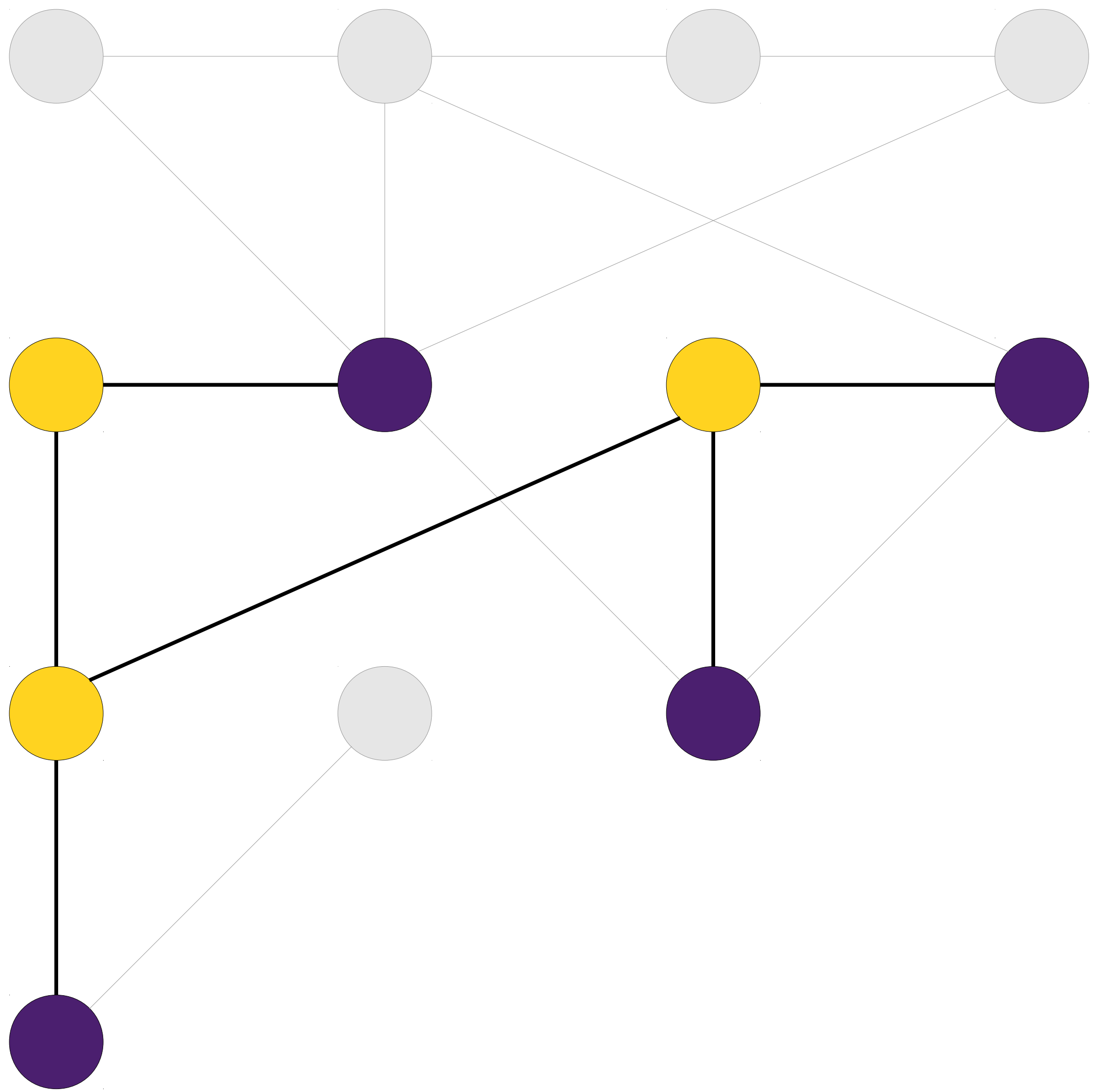
Iterating over a Graph



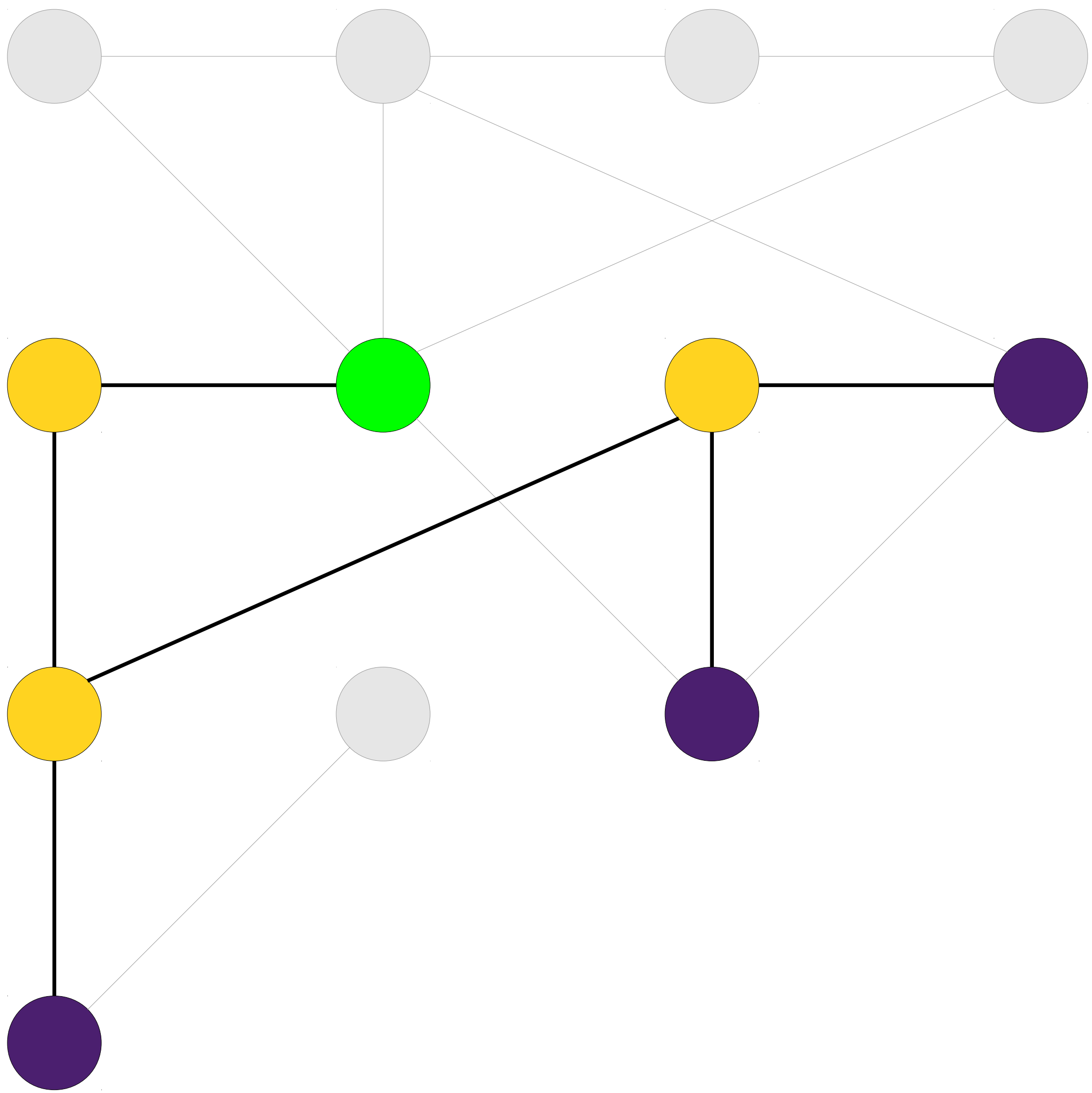
Iterating over a Graph



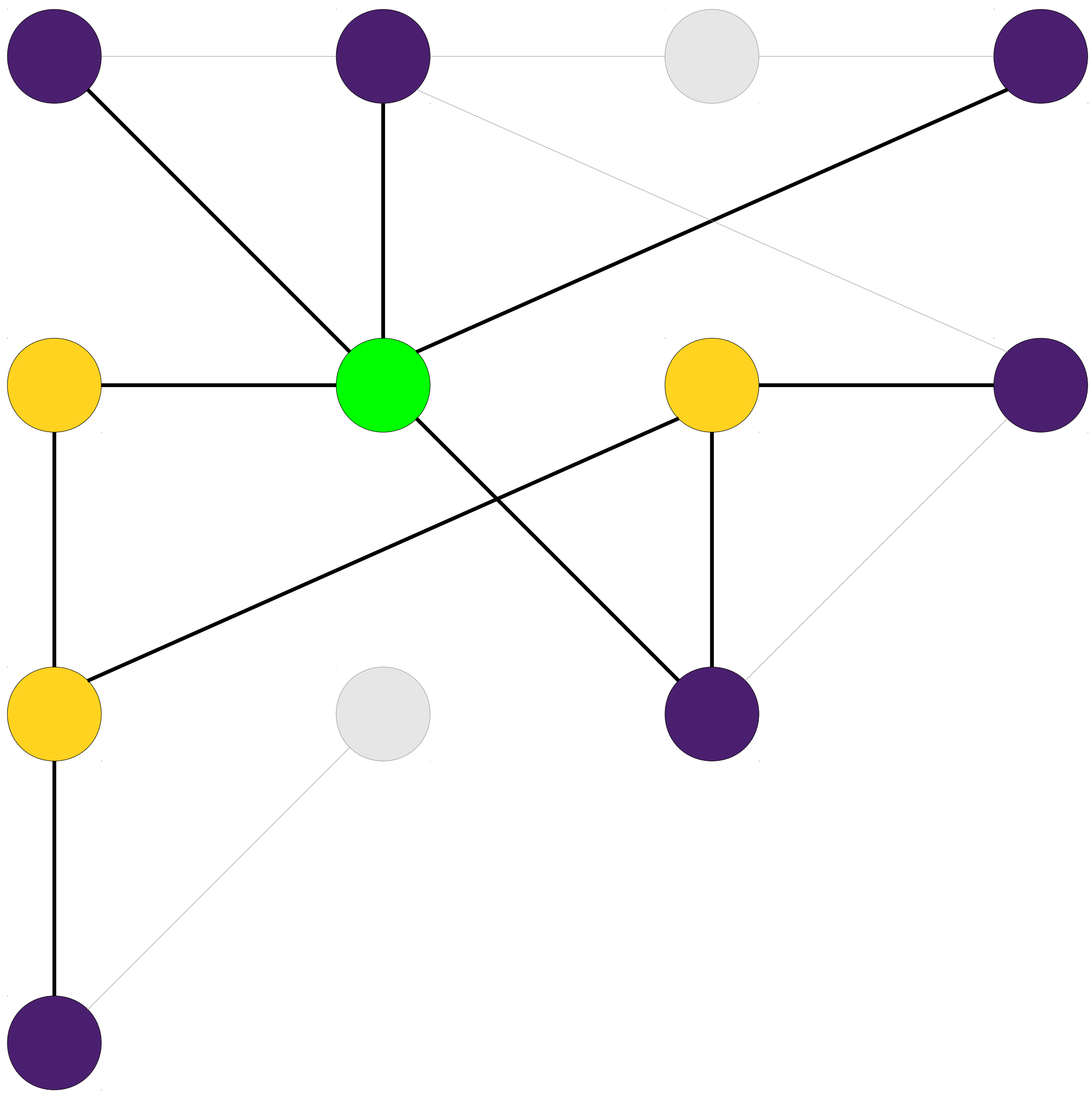
Iterating over a Graph



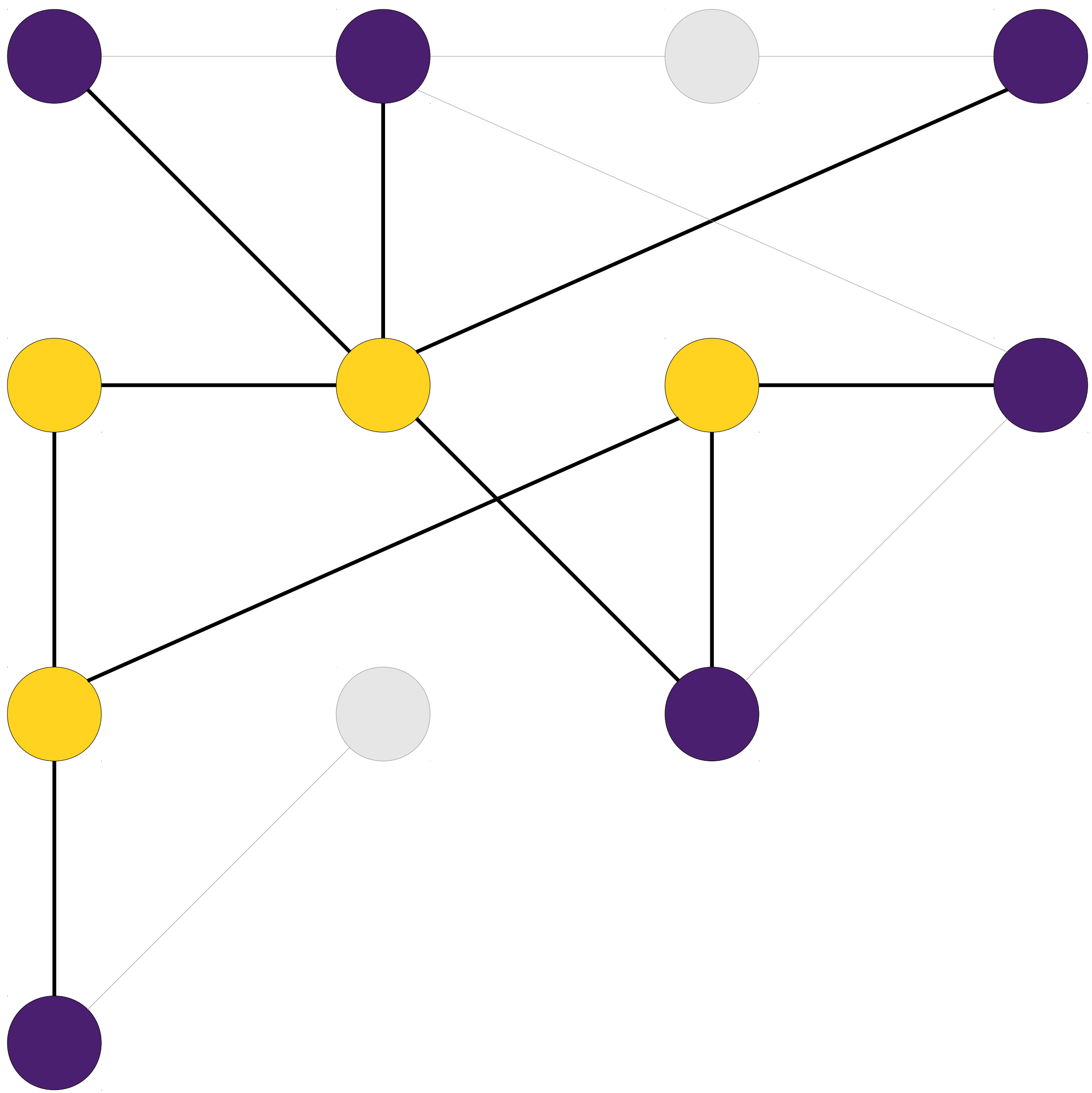
Iterating over a Graph



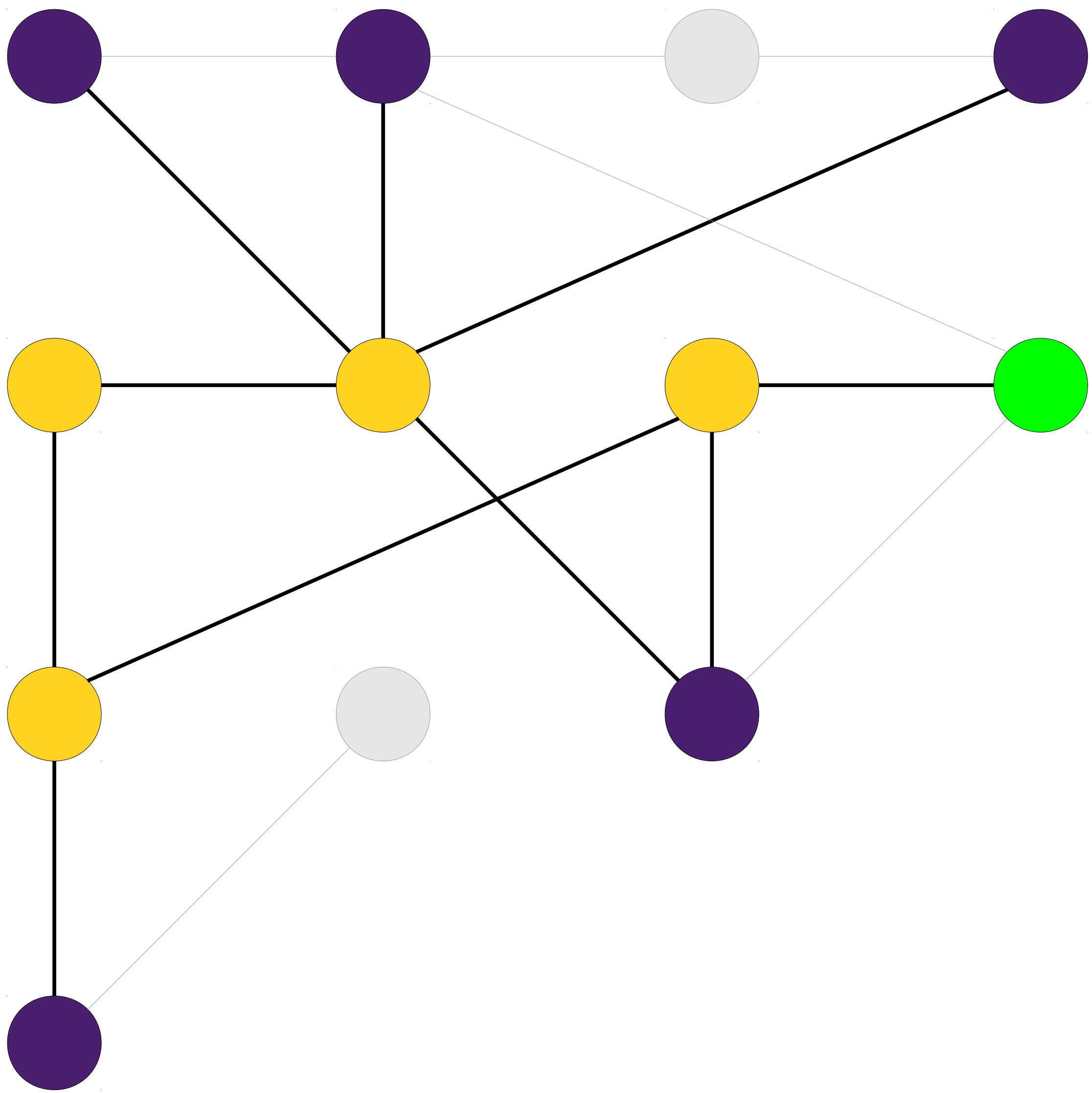
Iterating over a Graph



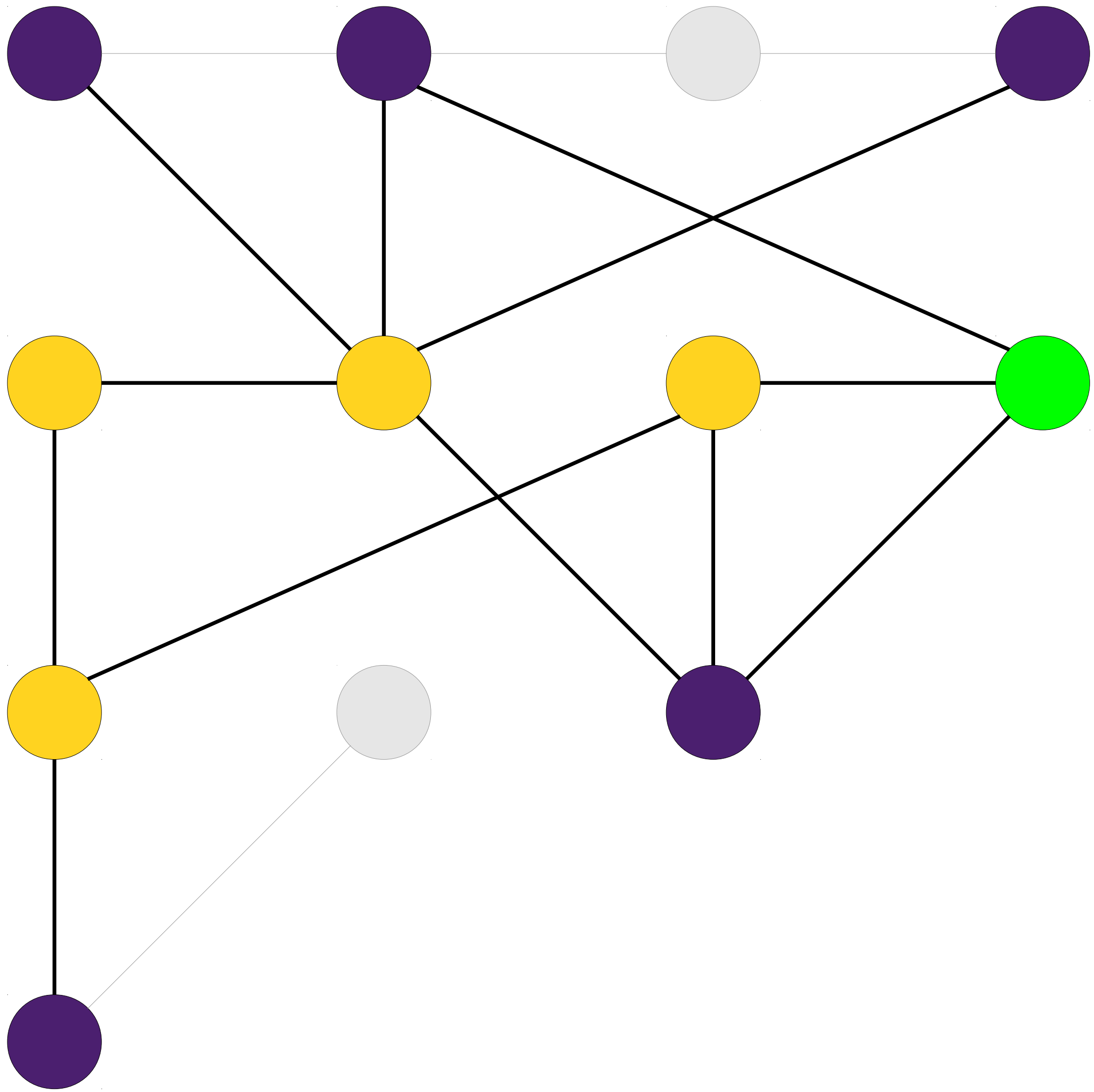
Iterating over a Graph



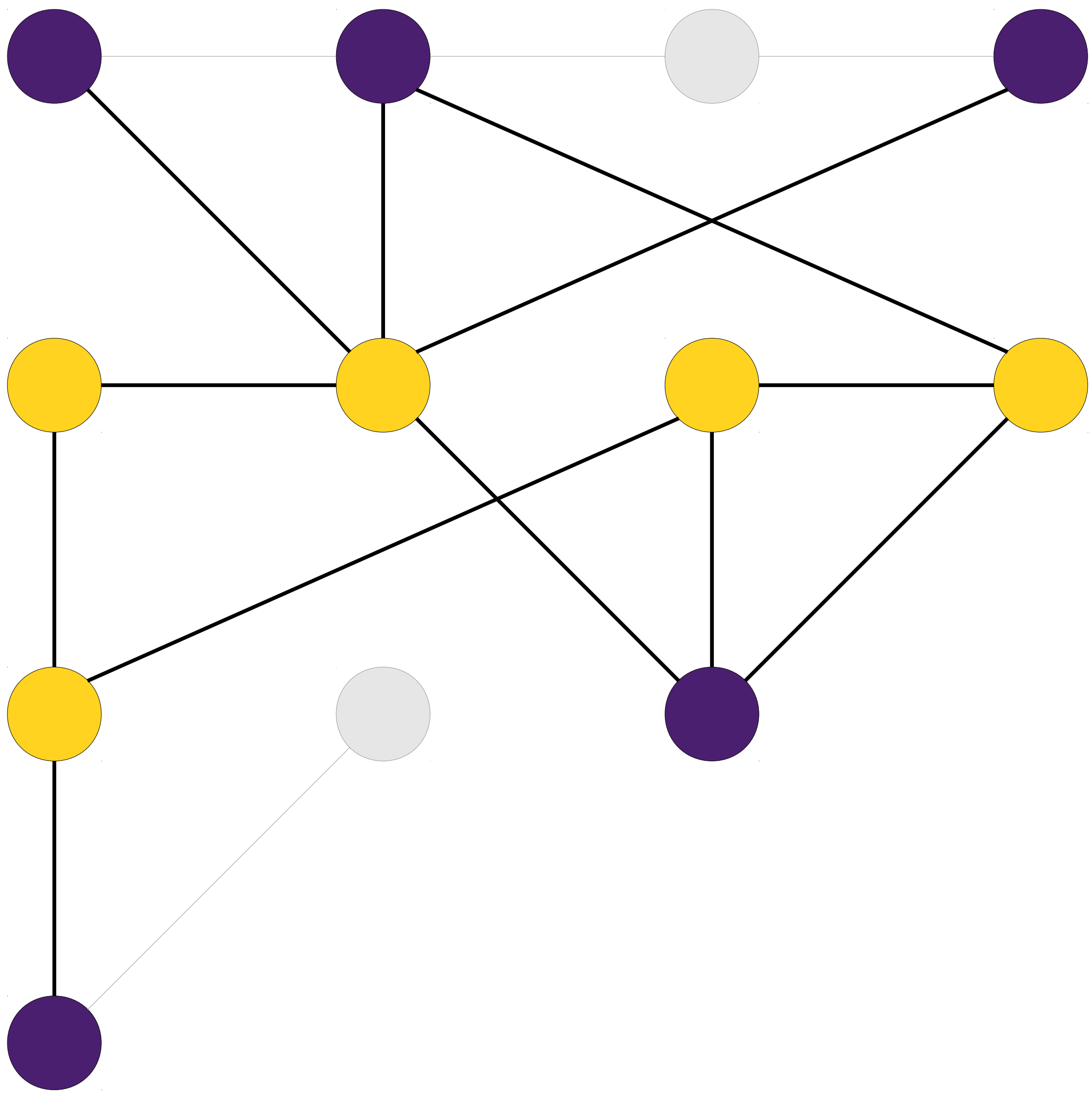
Iterating over a Graph



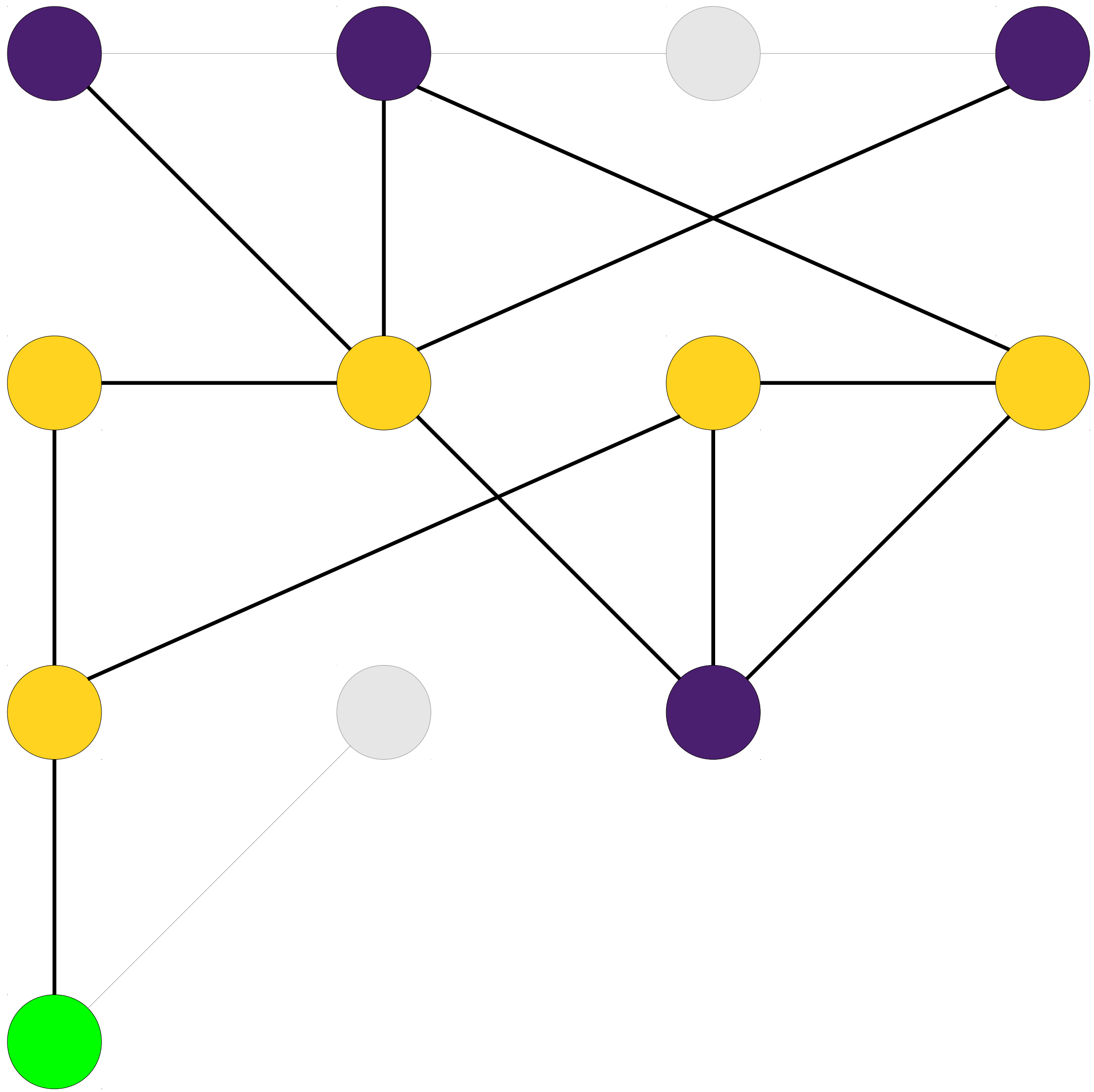
Iterating over a Graph



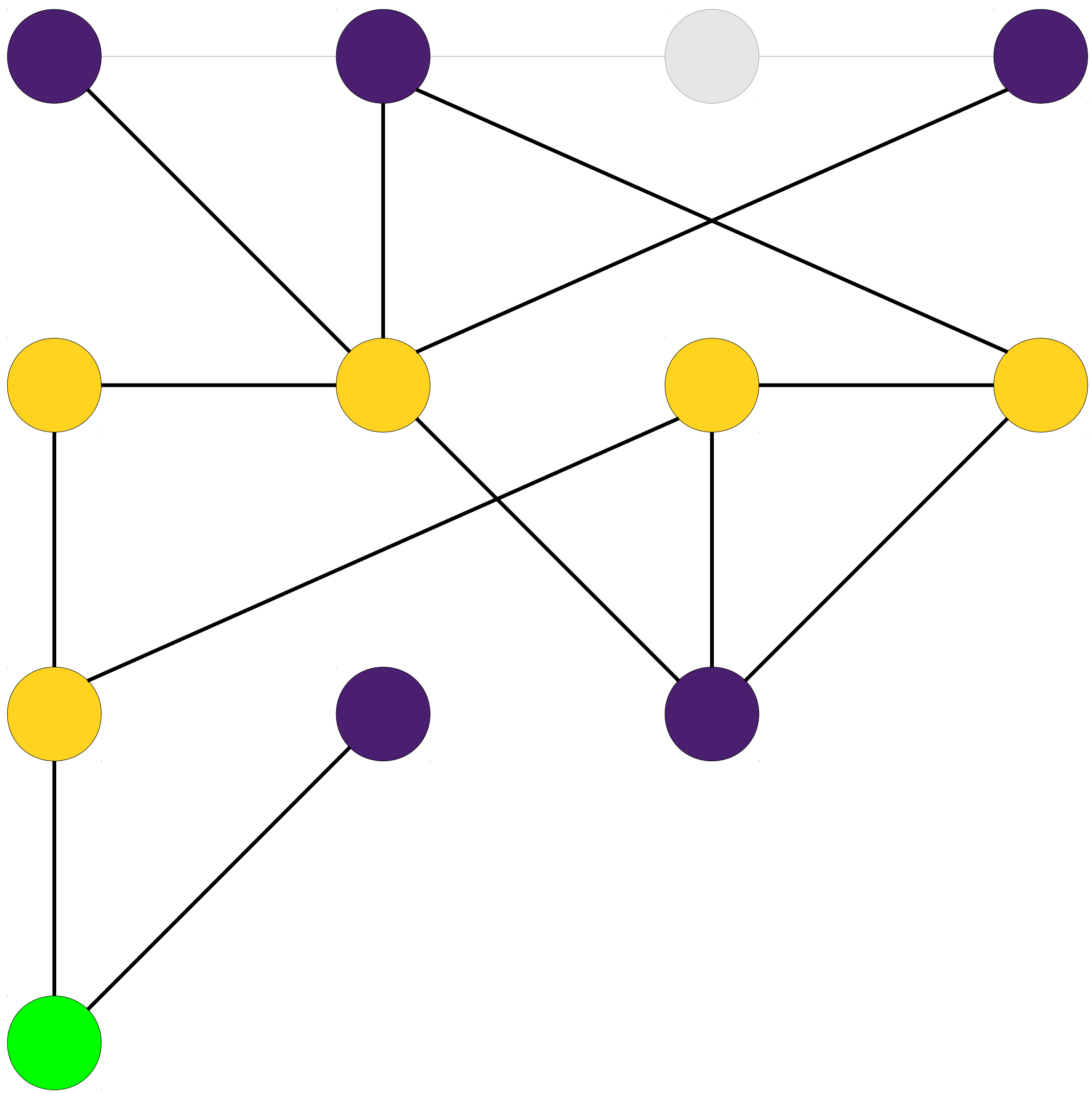
Iterating over a Graph



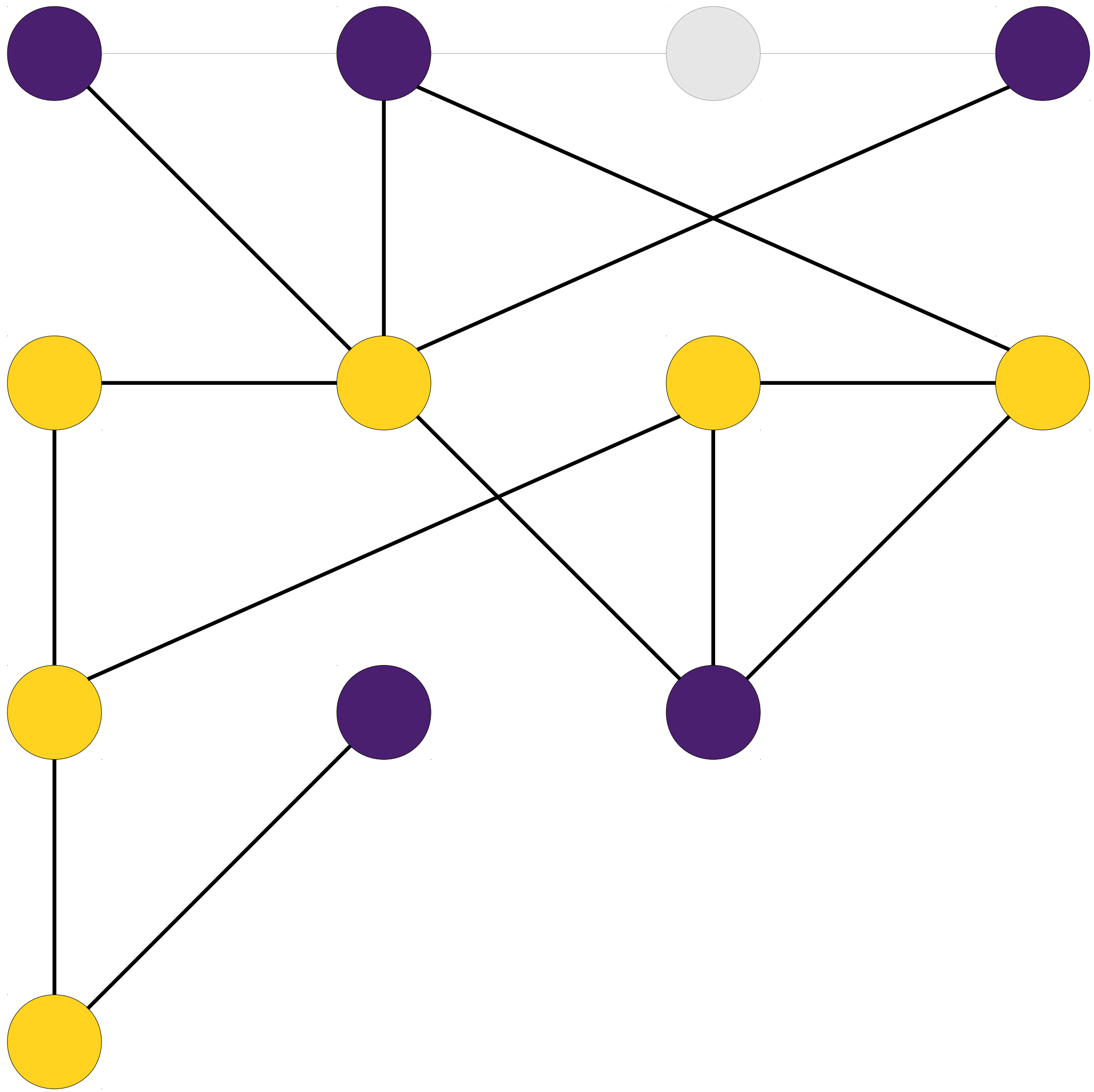
Iterating over a Graph



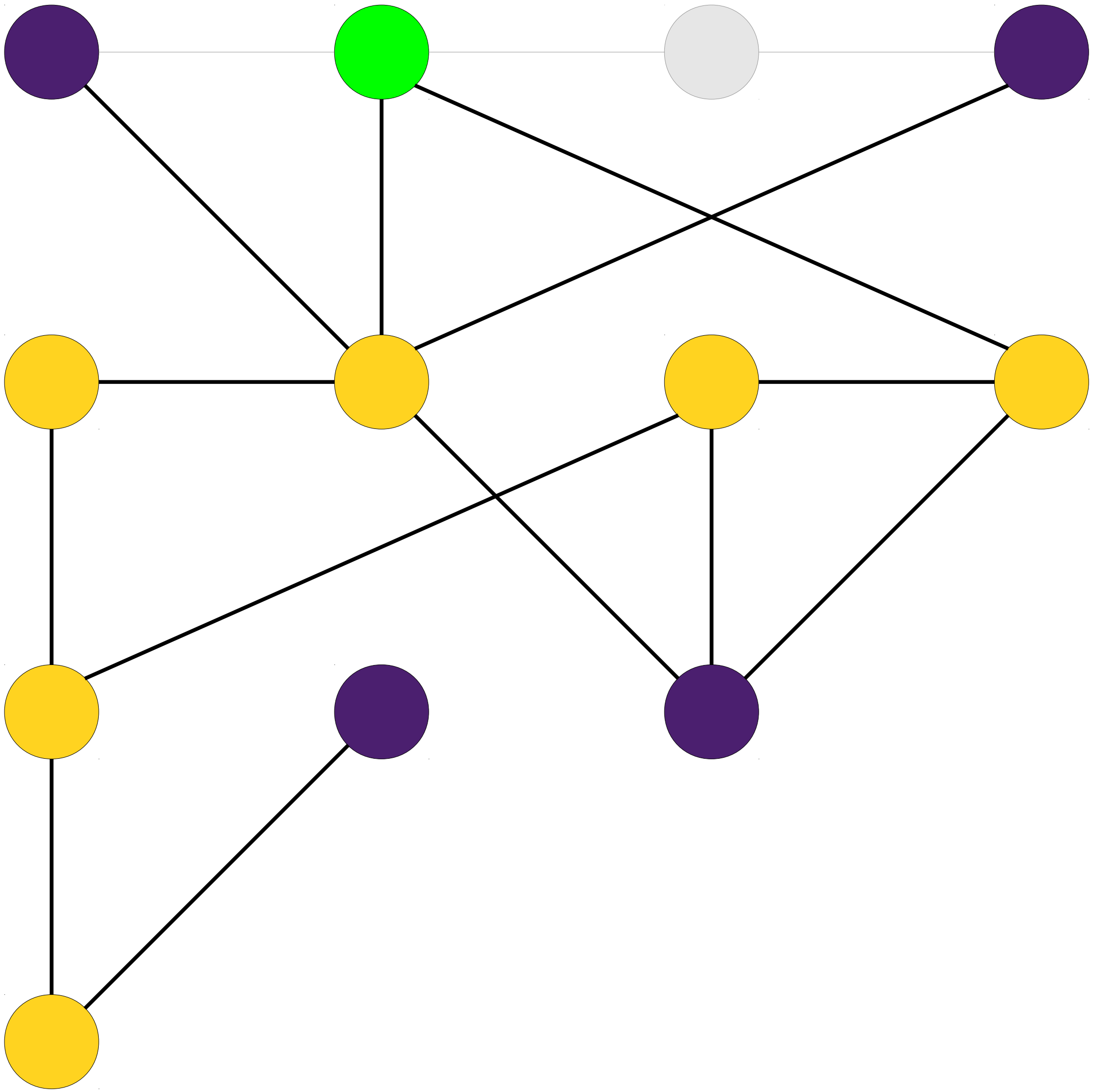
Iterating over a Graph



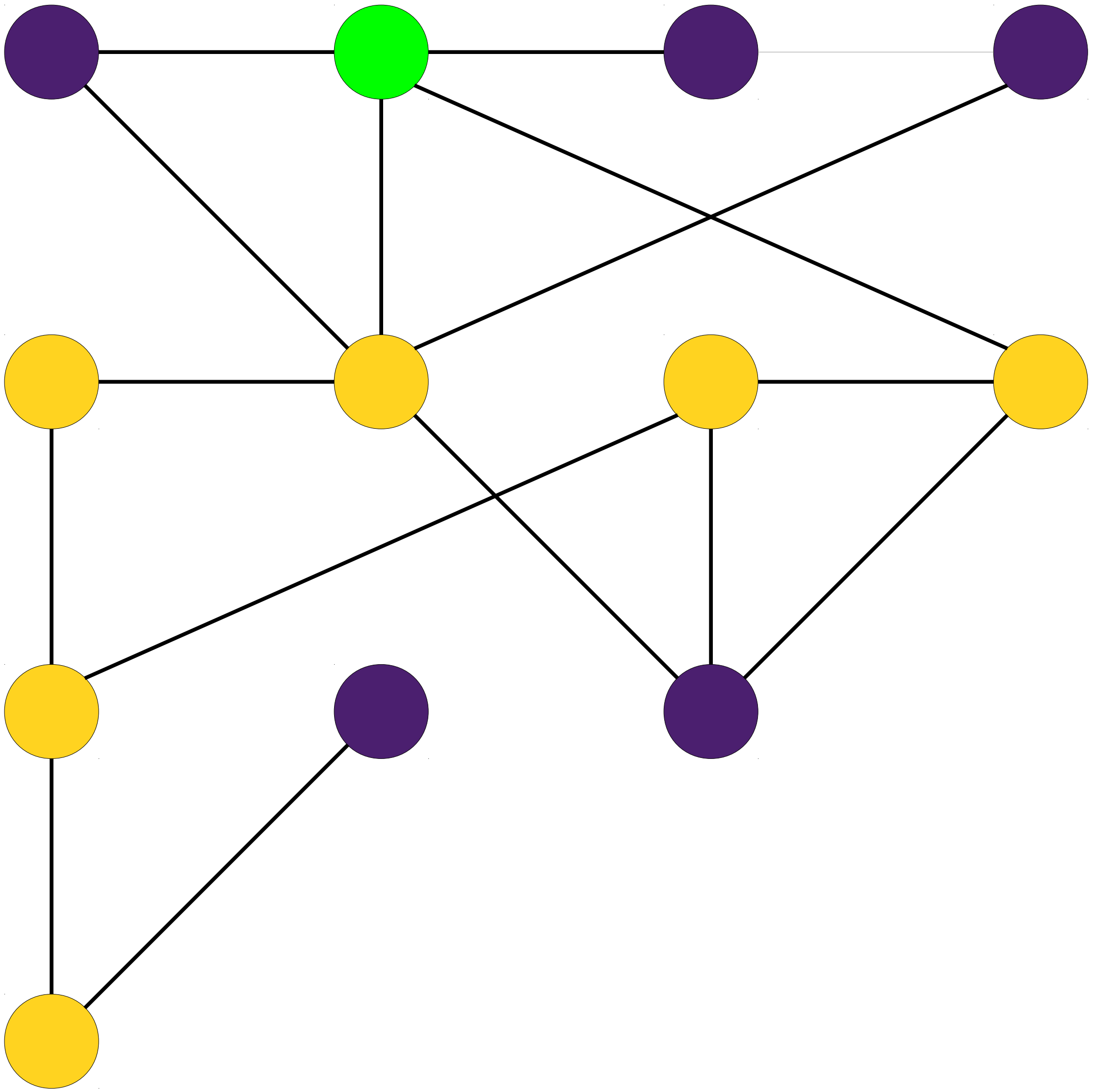
Iterating over a Graph



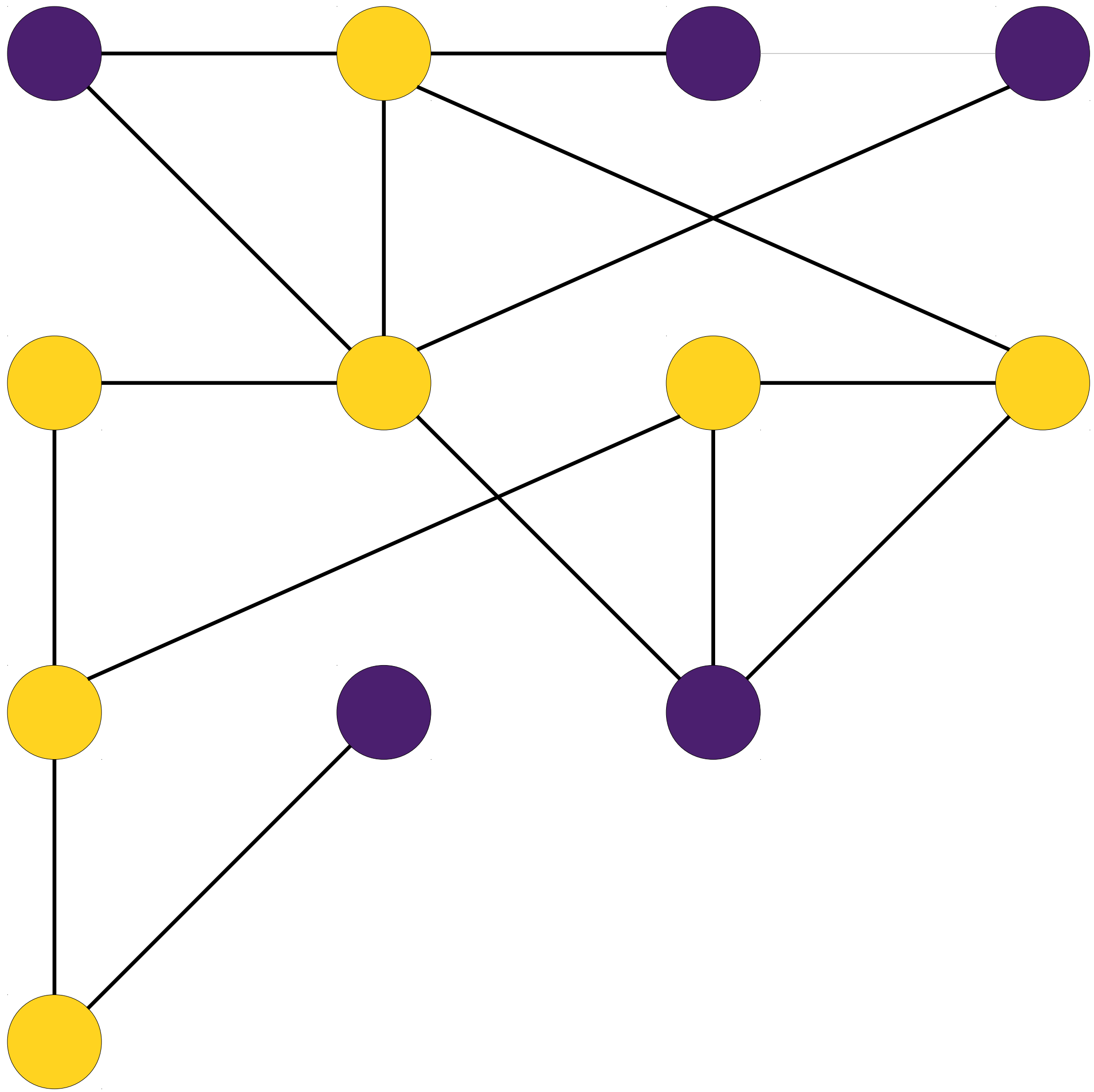
Iterating over a Graph



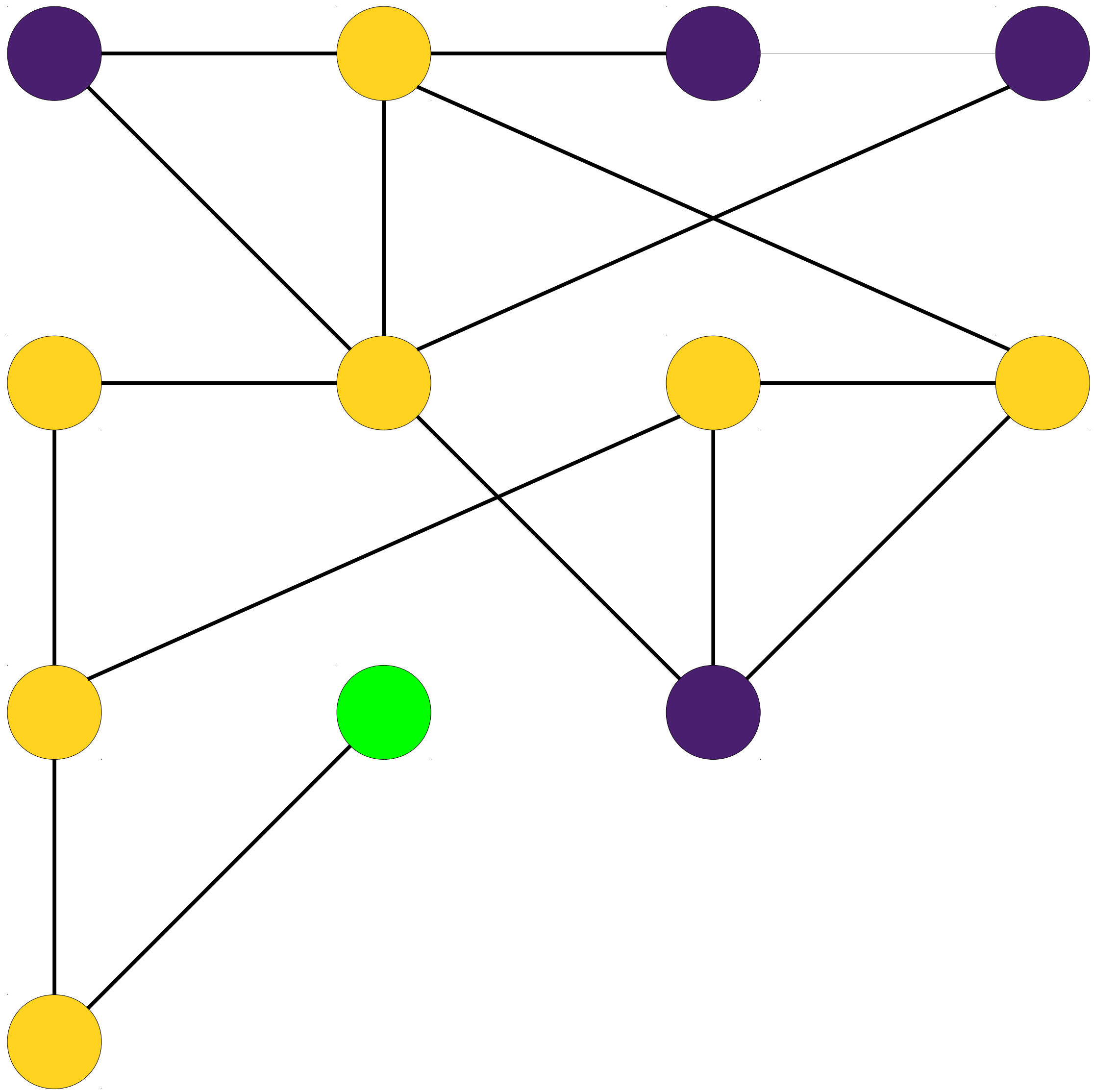
Iterating over a Graph



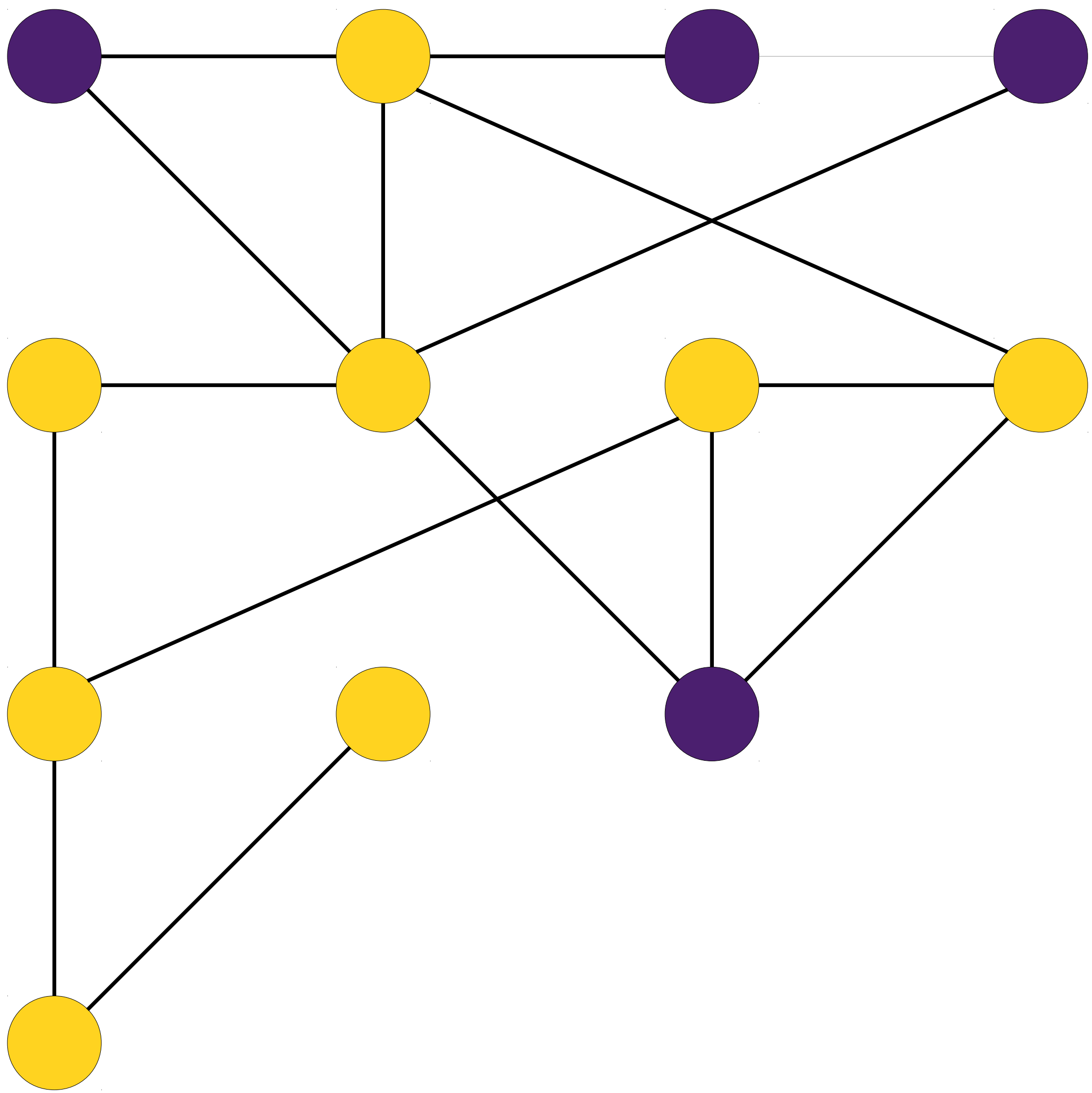
Iterating over a Graph



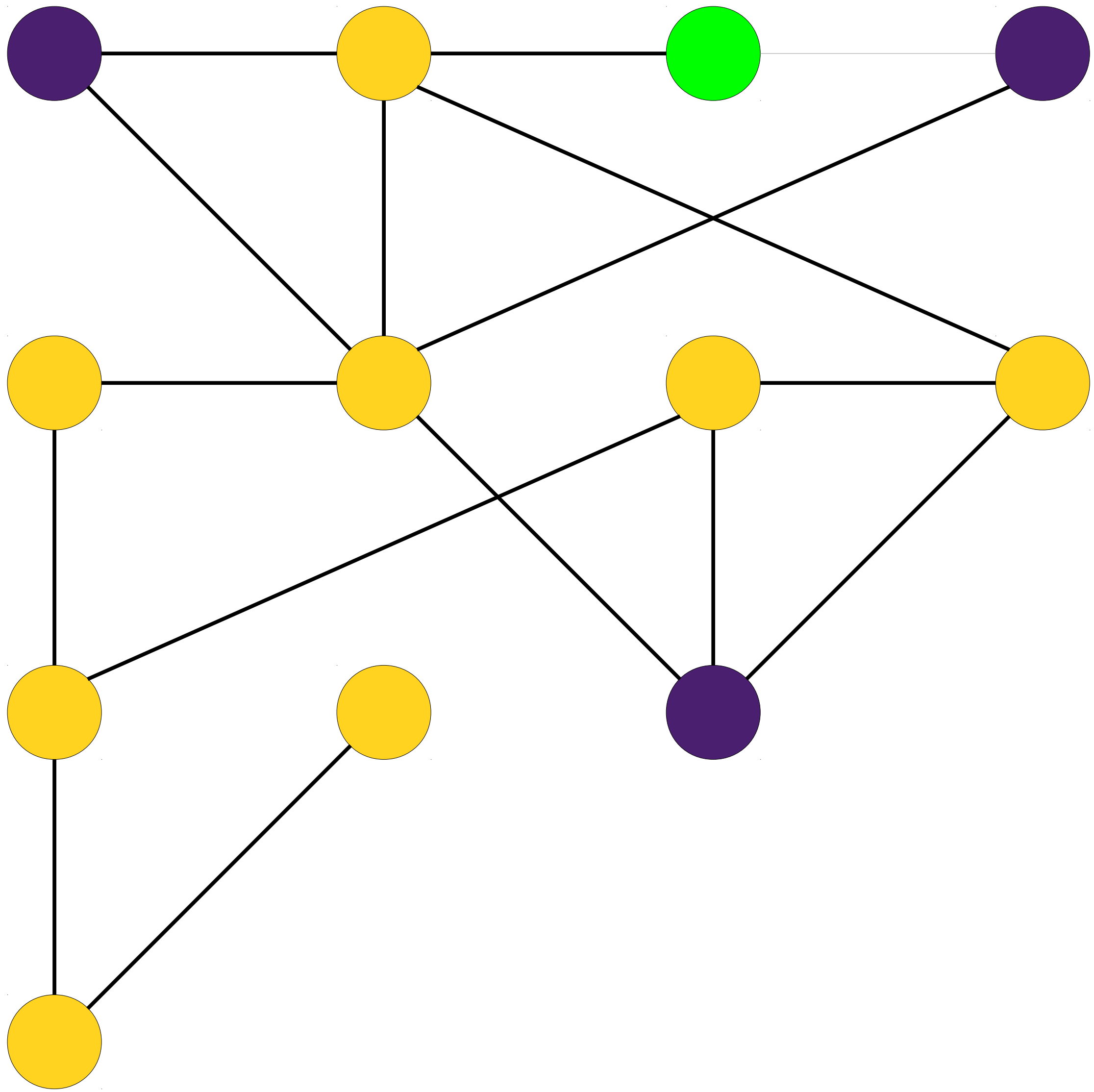
Iterating over a Graph



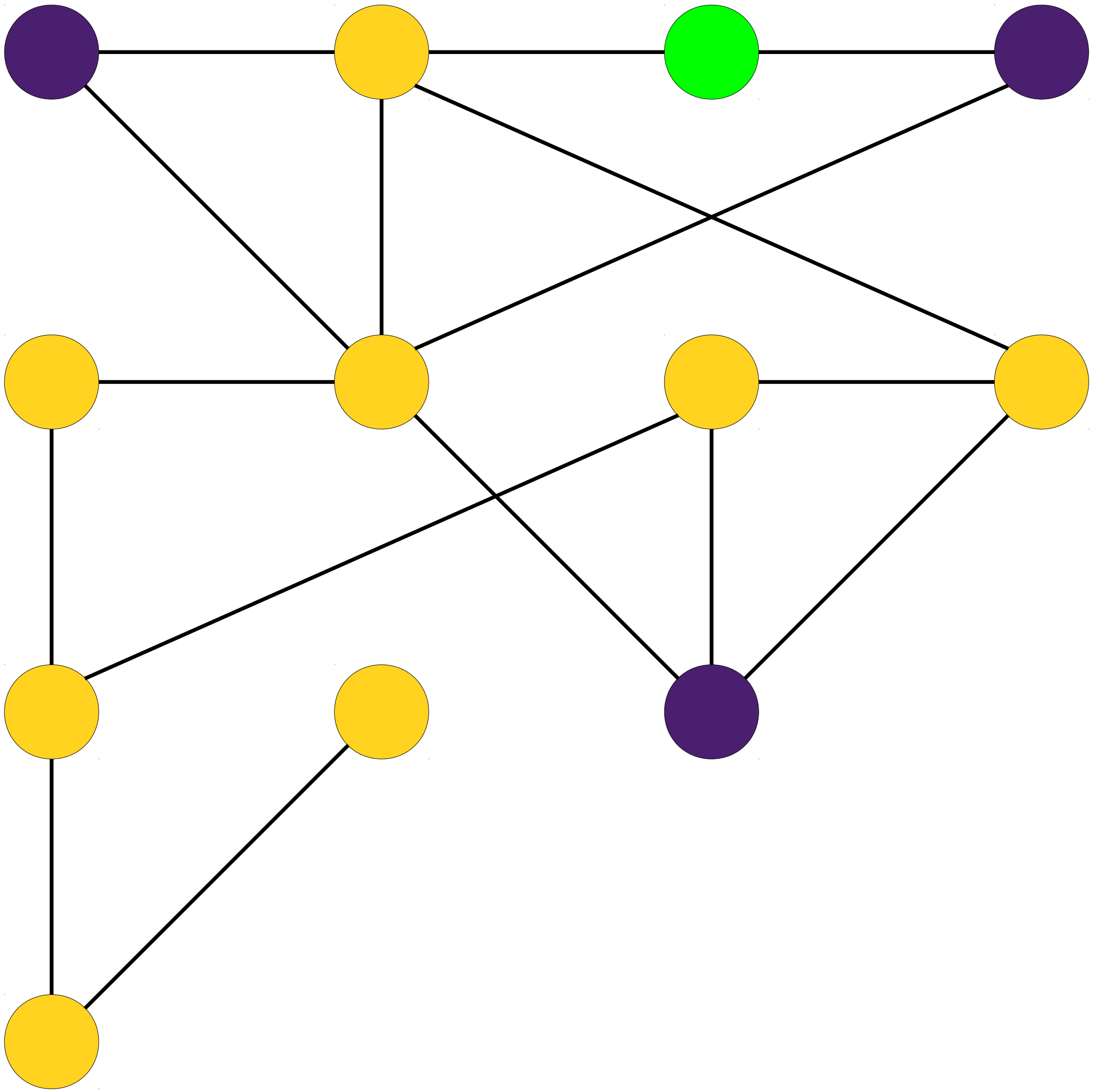
Iterating over a Graph



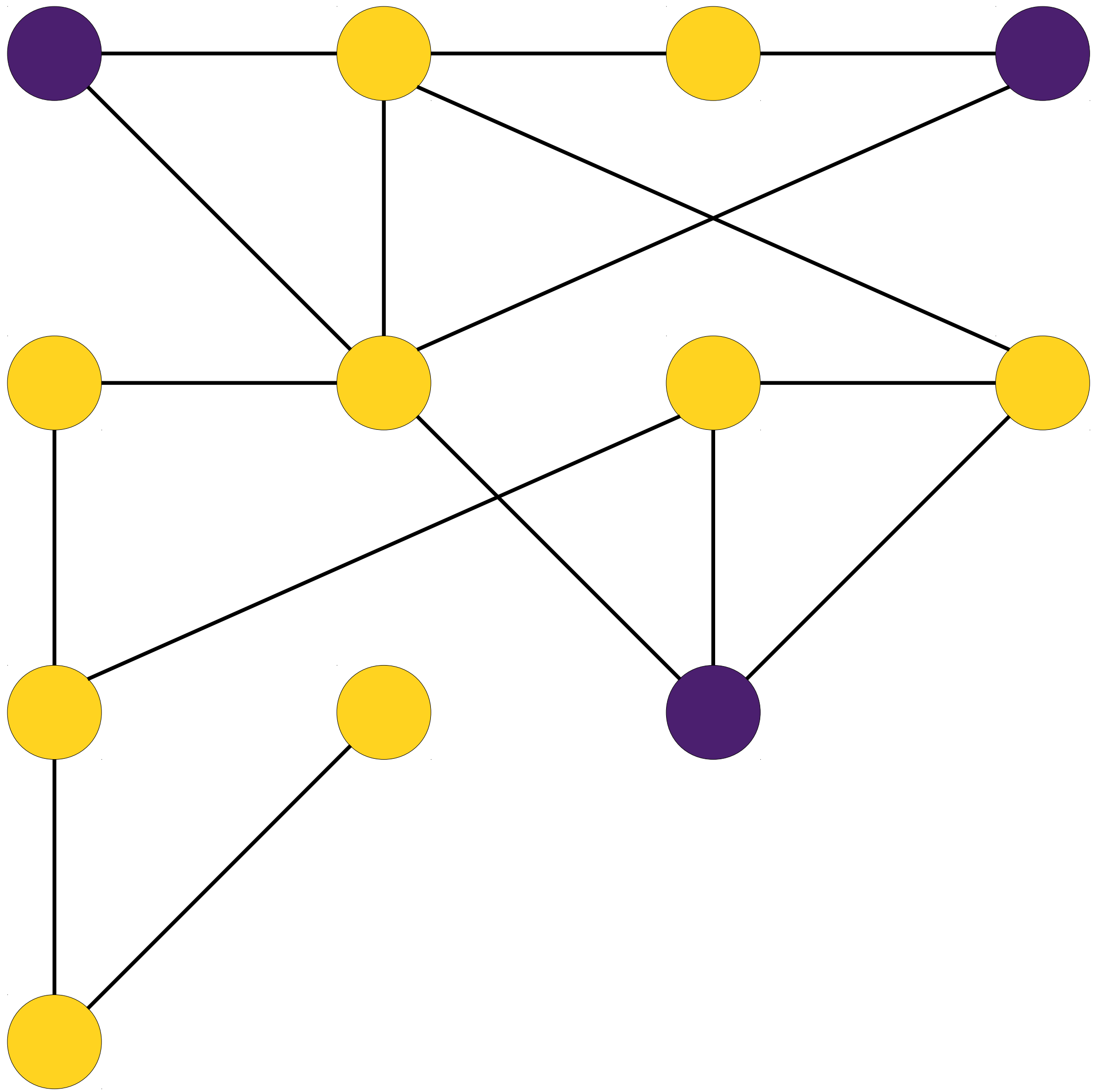
Iterating over a Graph



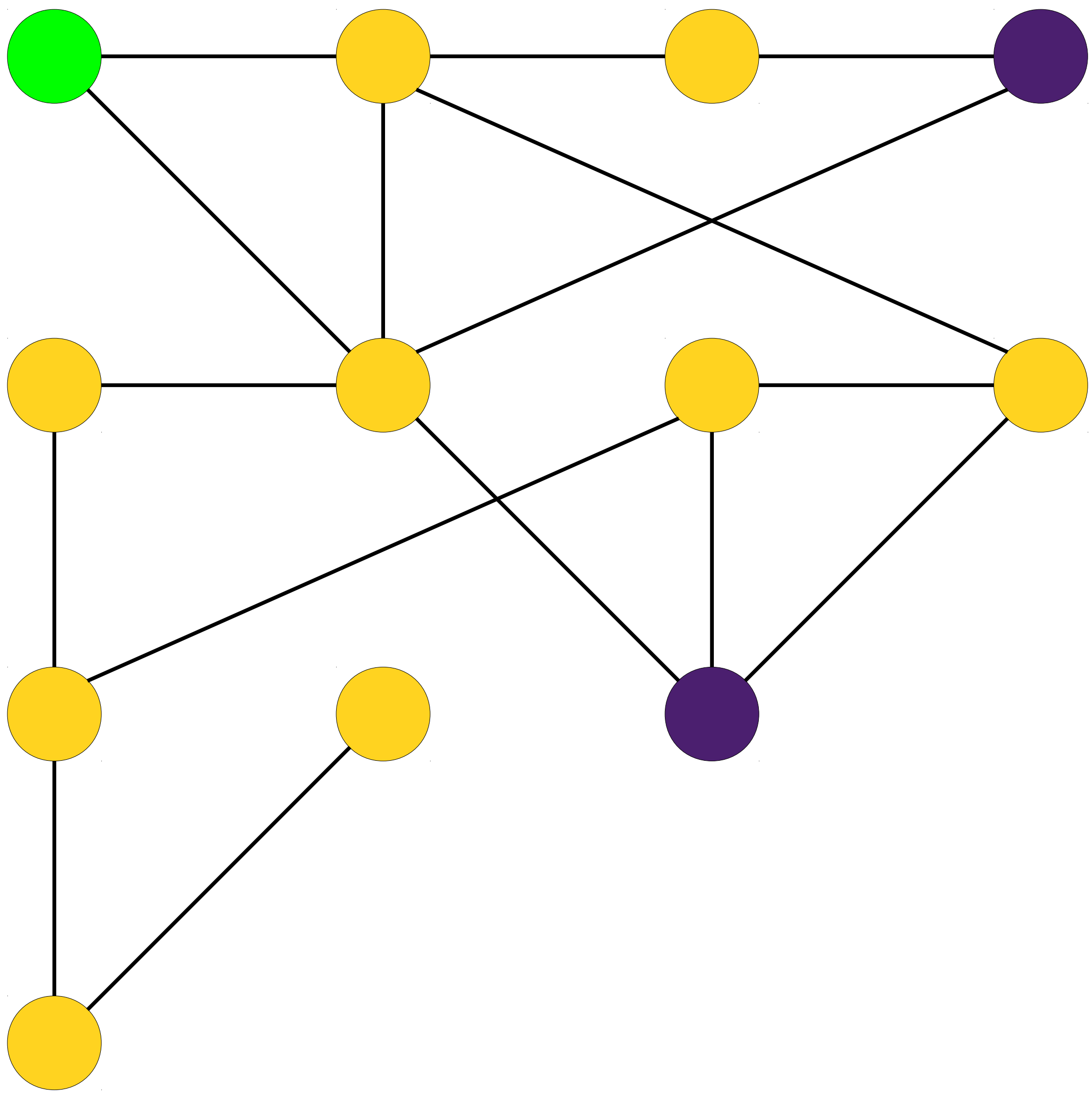
Iterating over a Graph



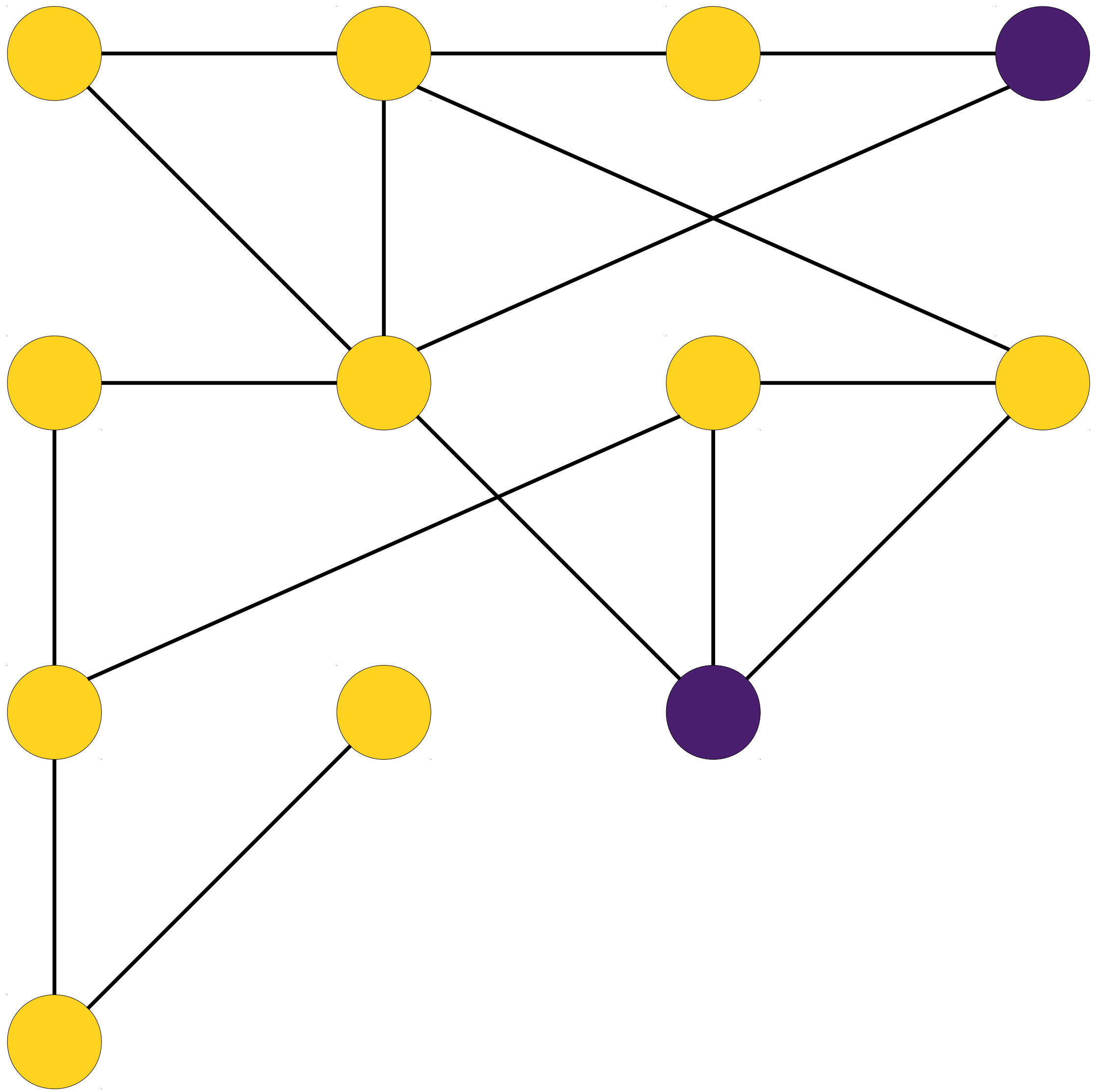
Iterating over a Graph



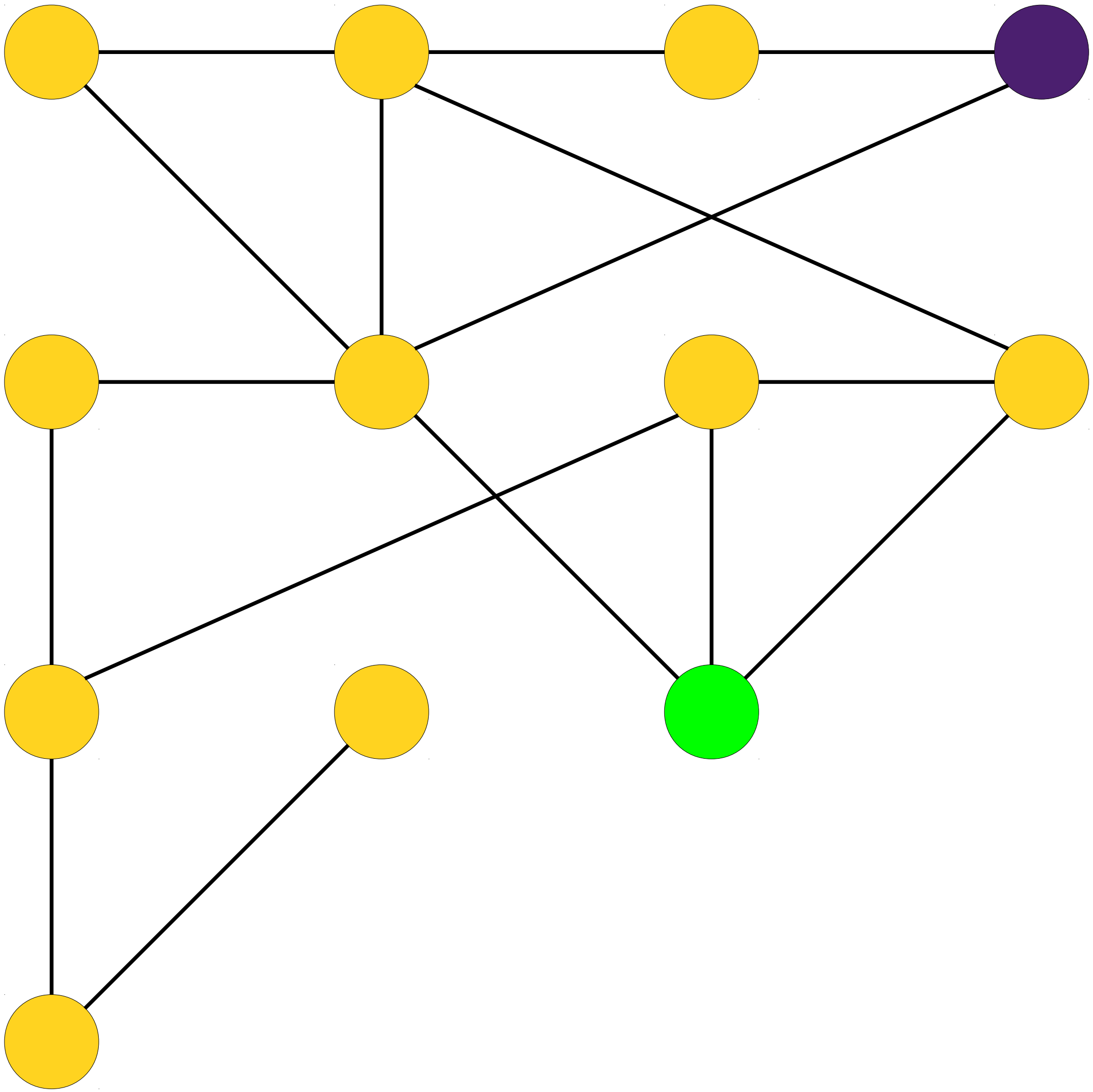
Iterating over a Graph



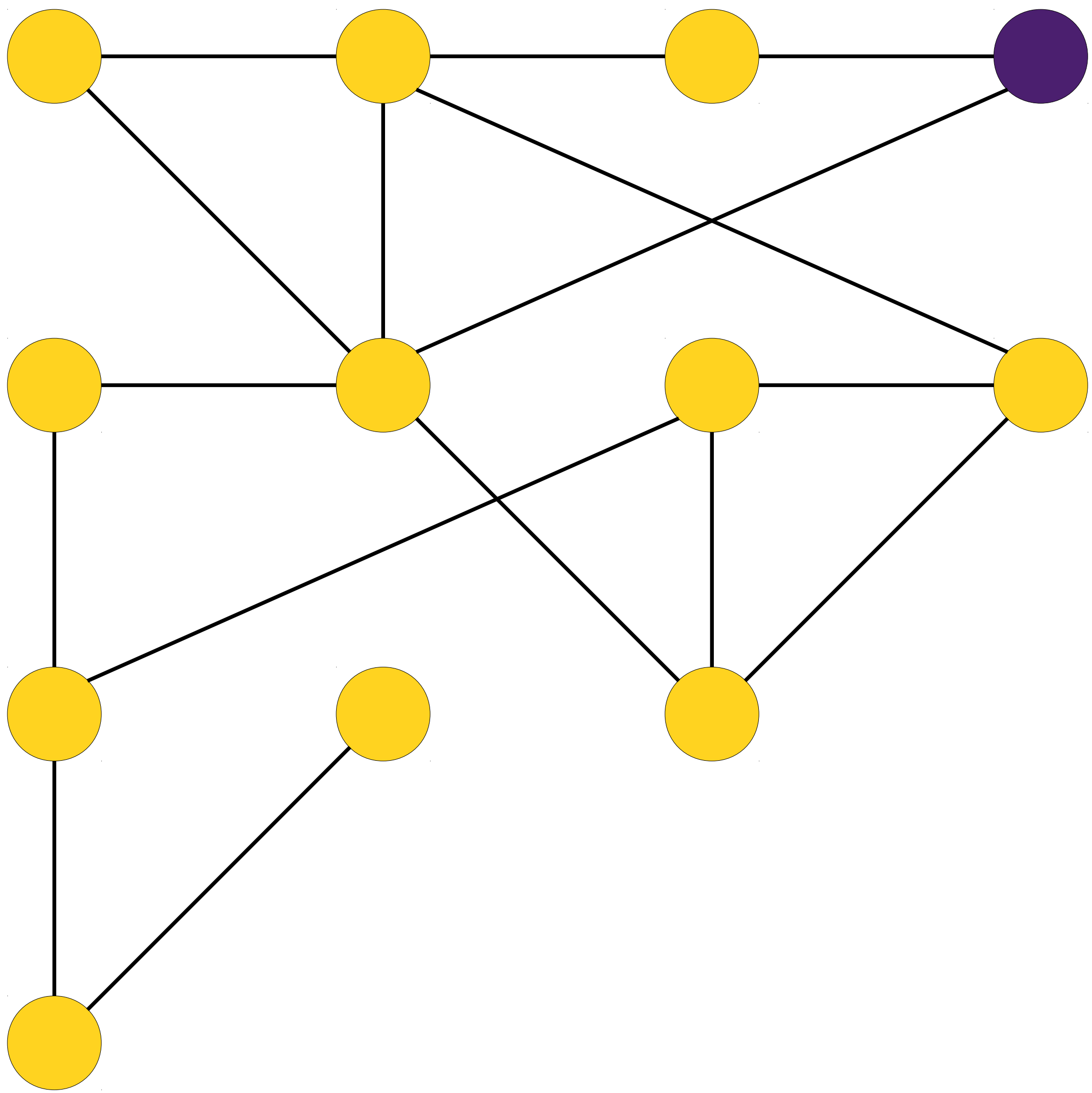
Iterating over a Graph



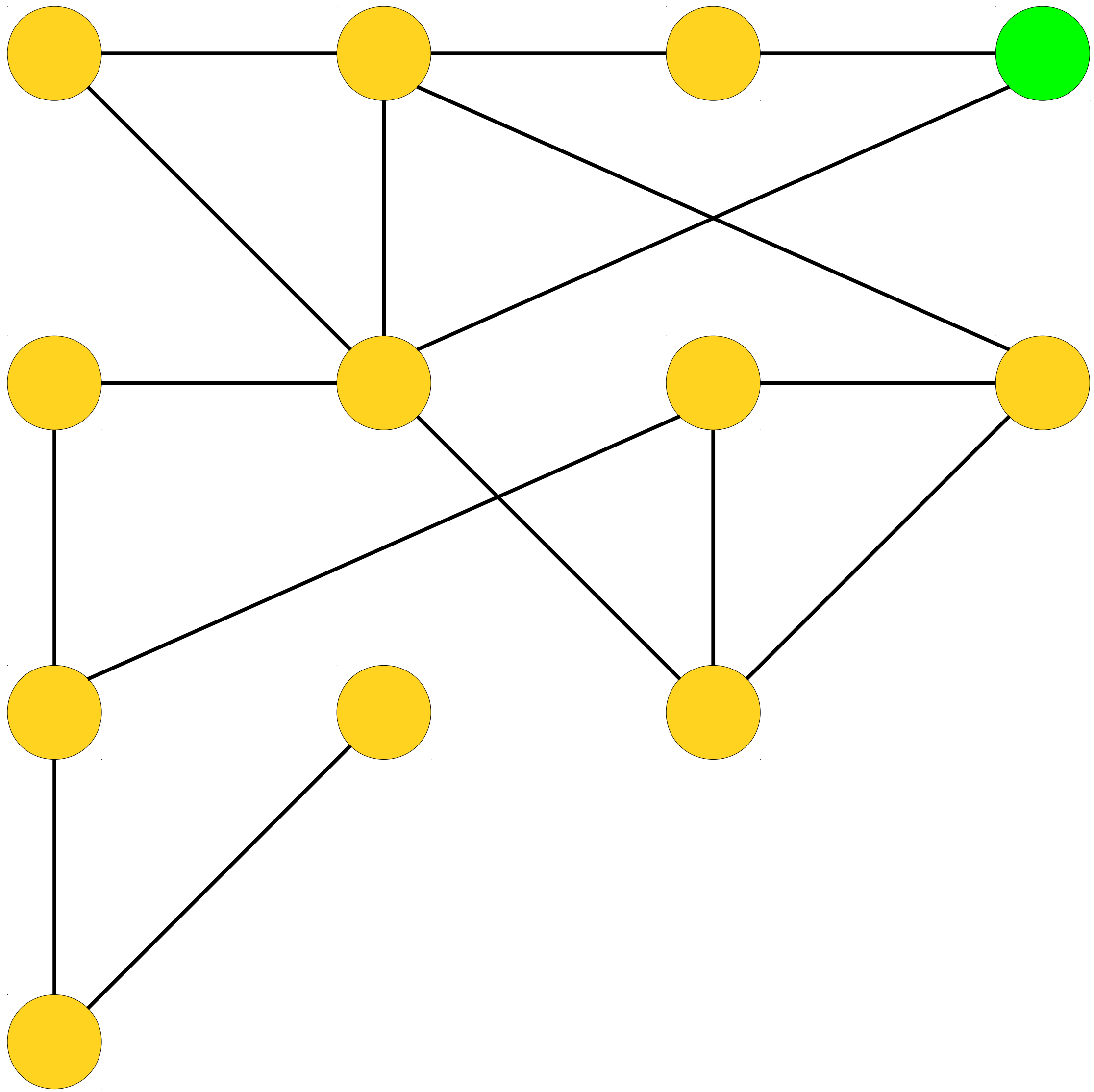
Iterating over a Graph



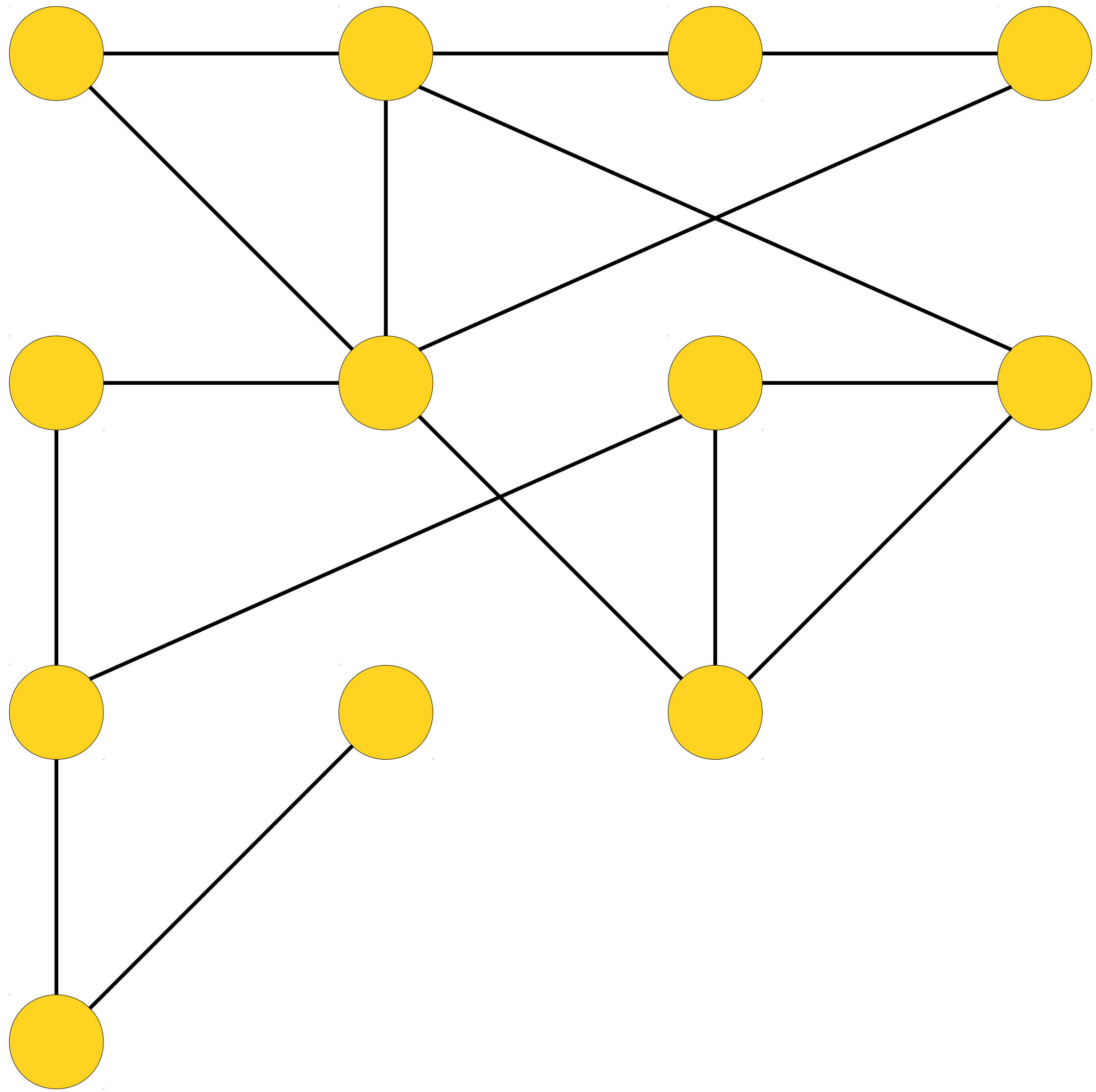
Iterating over a Graph



Iterating over a Graph



Iterating over a Graph



Iterating over a Graph

- Maintain a collection C of nodes to visit.
- Initialize C with a start node.
- While C is not empty:
  - Pick a node v out of C.
  - Follow all outgoing edges from v, adding each unvisited node found this way to C.
- Eventually explores all nodes reachable from the starting set of nodes.

Iterating over a Graph

Maintain a collection C of nodes to visit.

Initialize C with a start node.

While C is not empty:

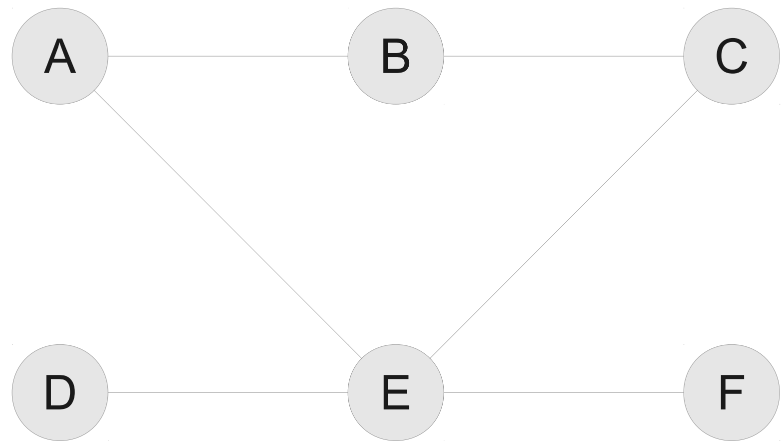
- Pick a node v out of C.

  Follow all outgoing edges from v, adding each unvisited node found this way to C.

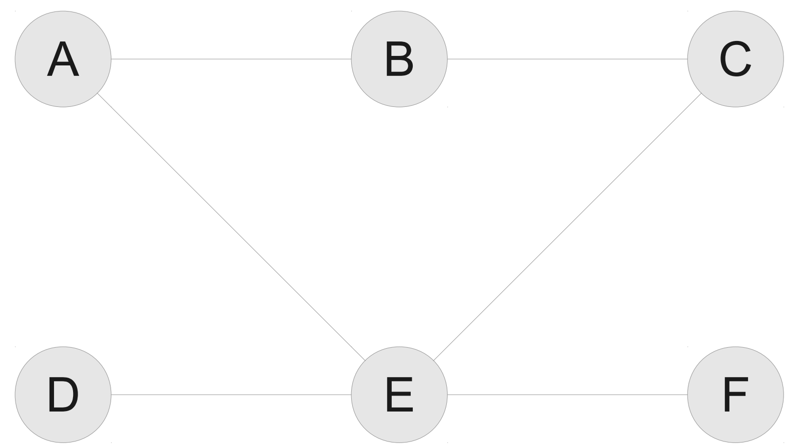
Eventually explores all nodes reachable from the starting set of nodes.

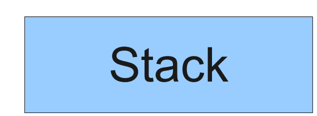
Depth-First Search

Depth-first search

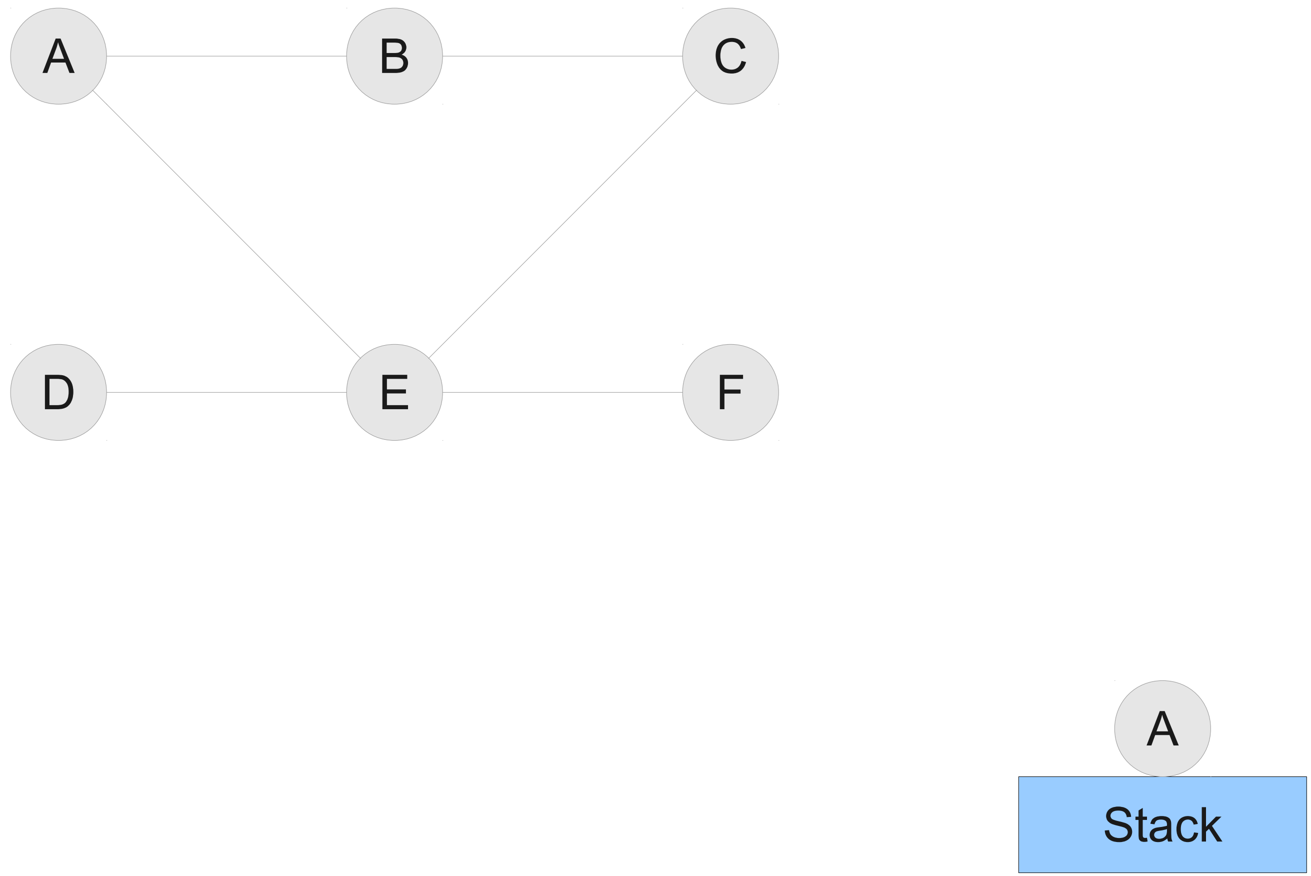


Depth-first search

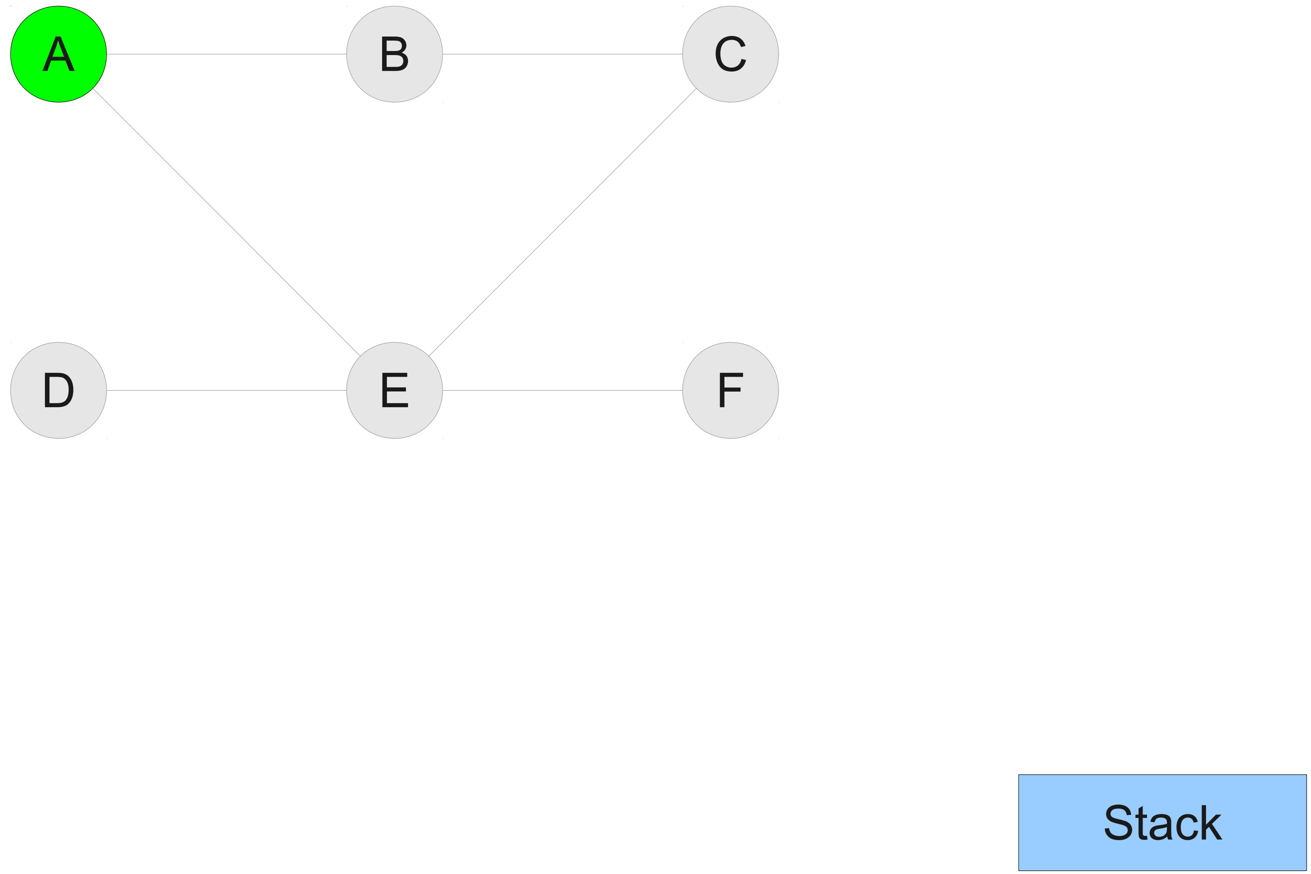




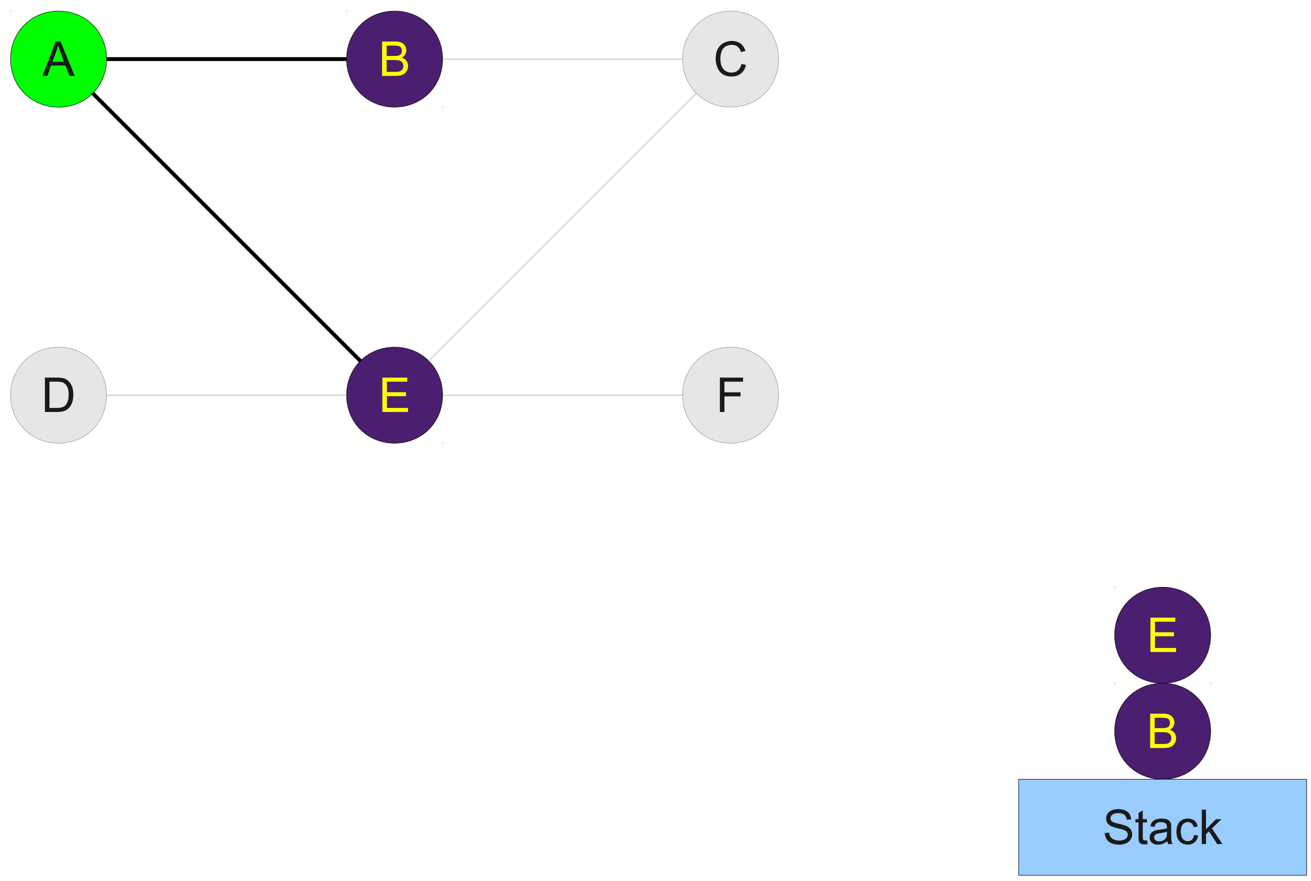
Depth-first search



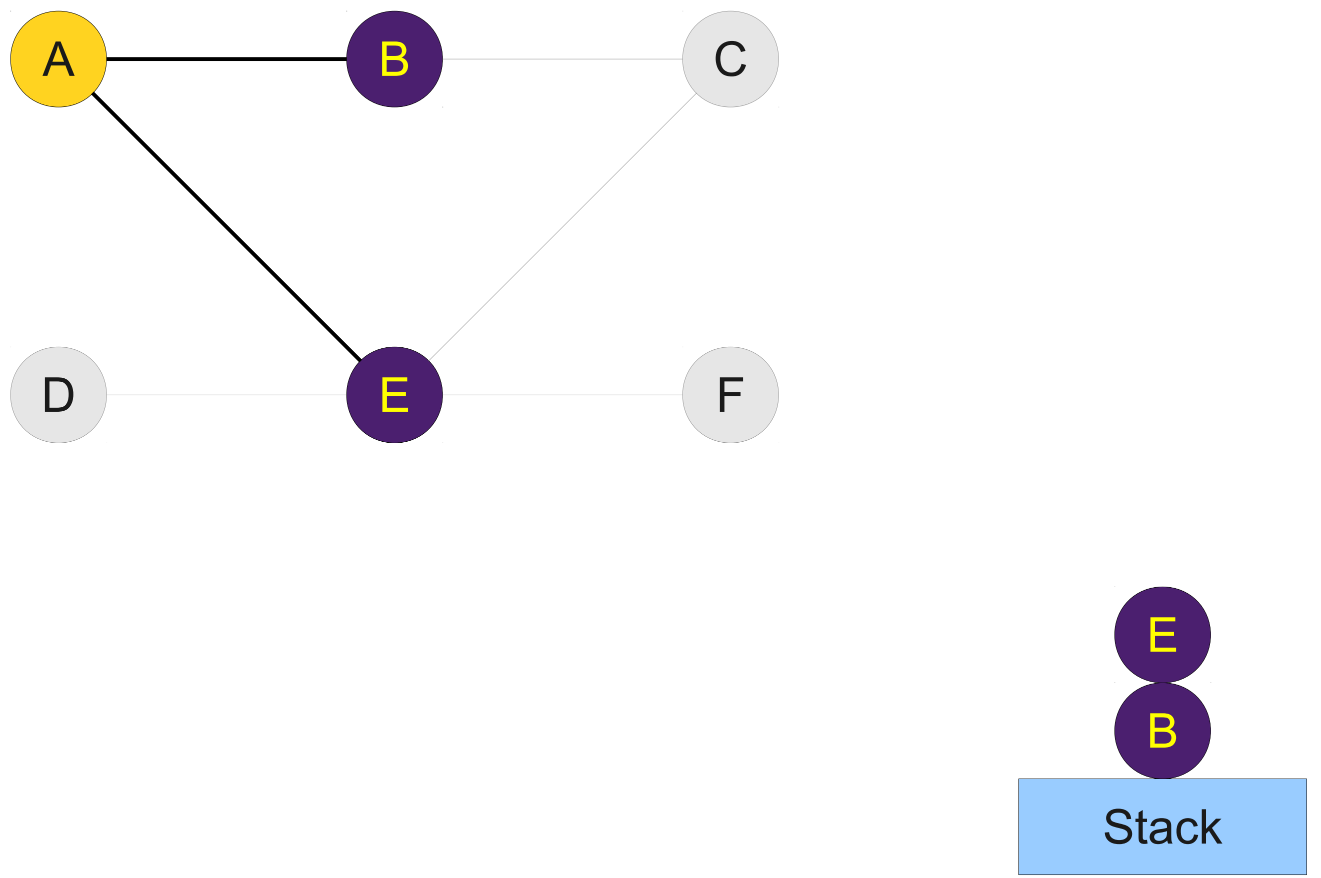
Depth-first search



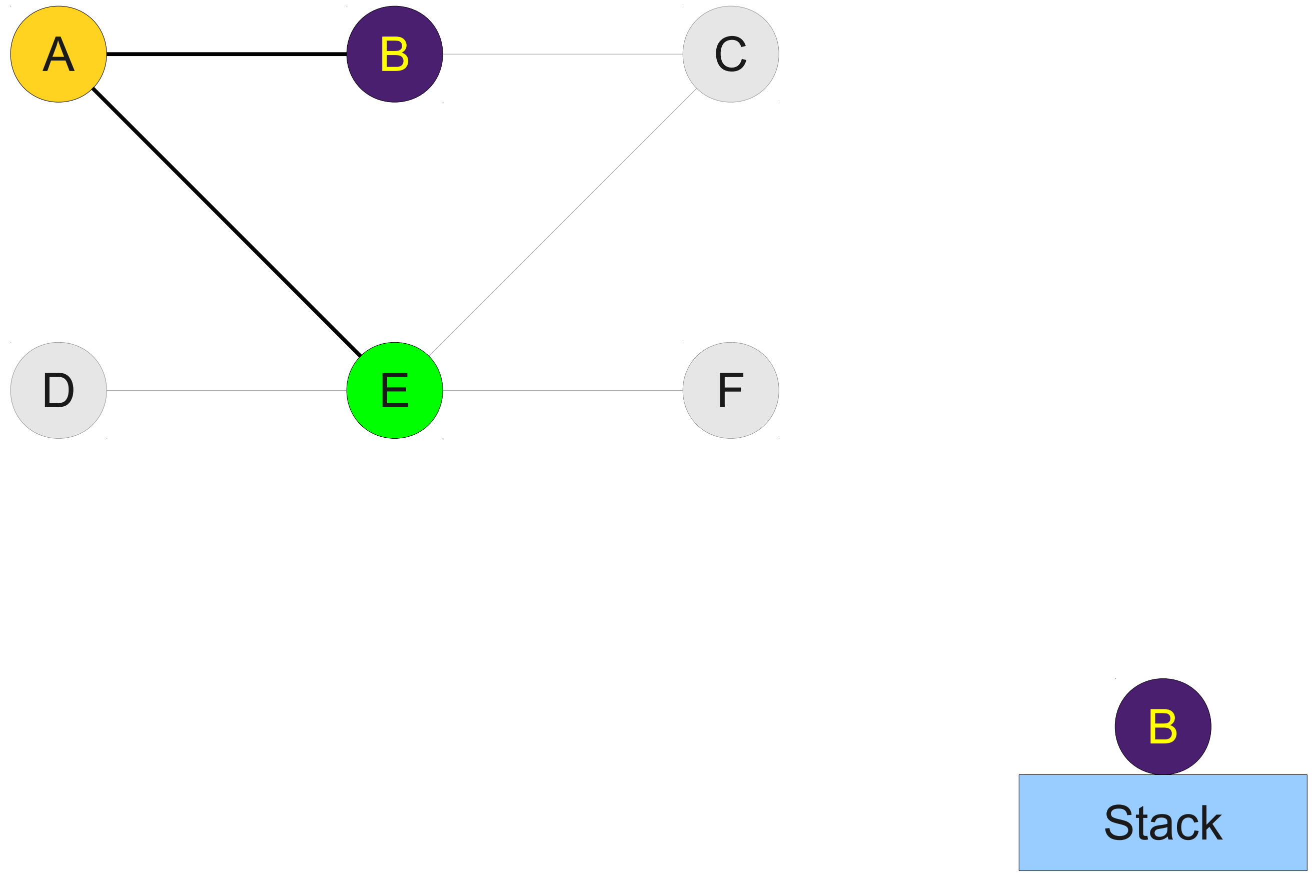
Depth-first search



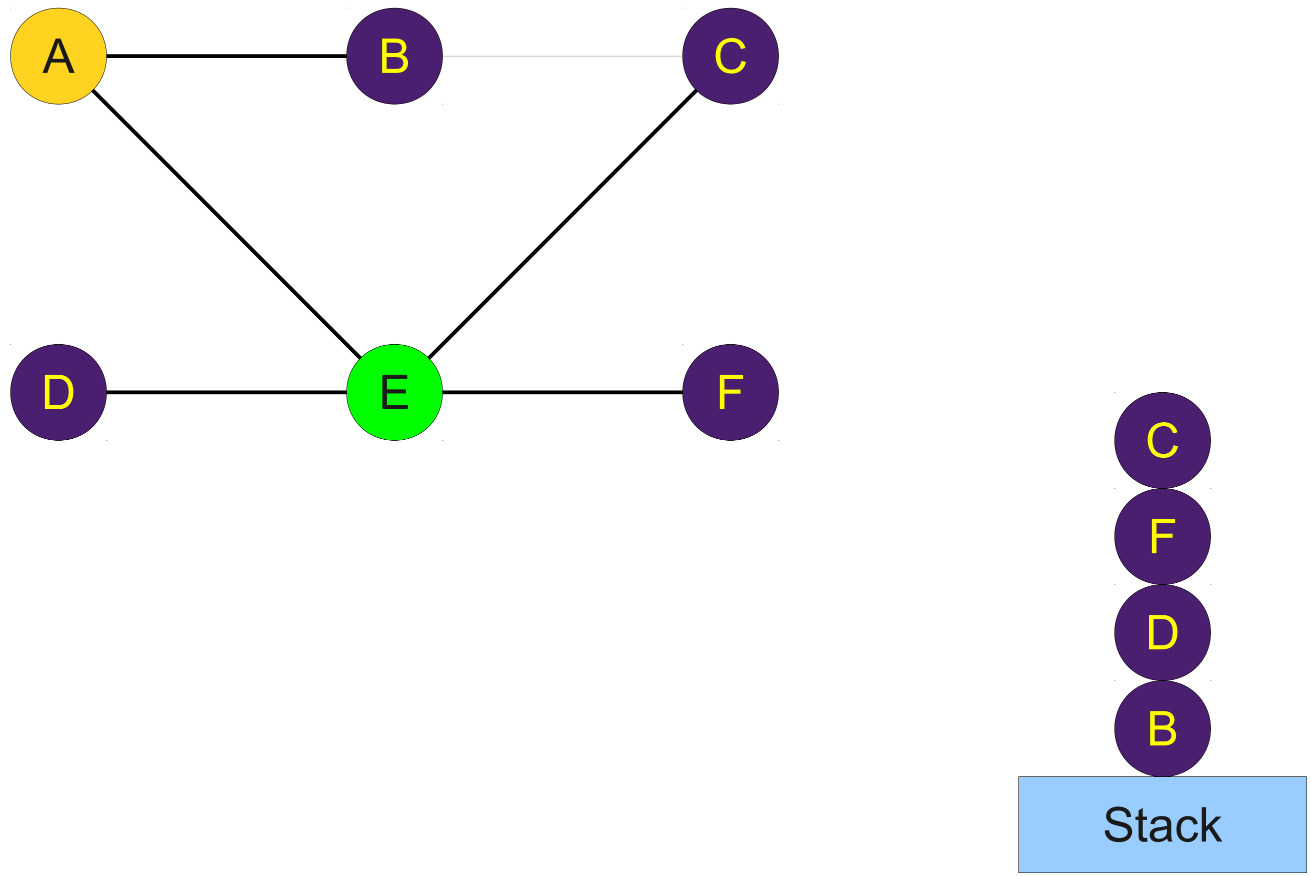
Depth-first search



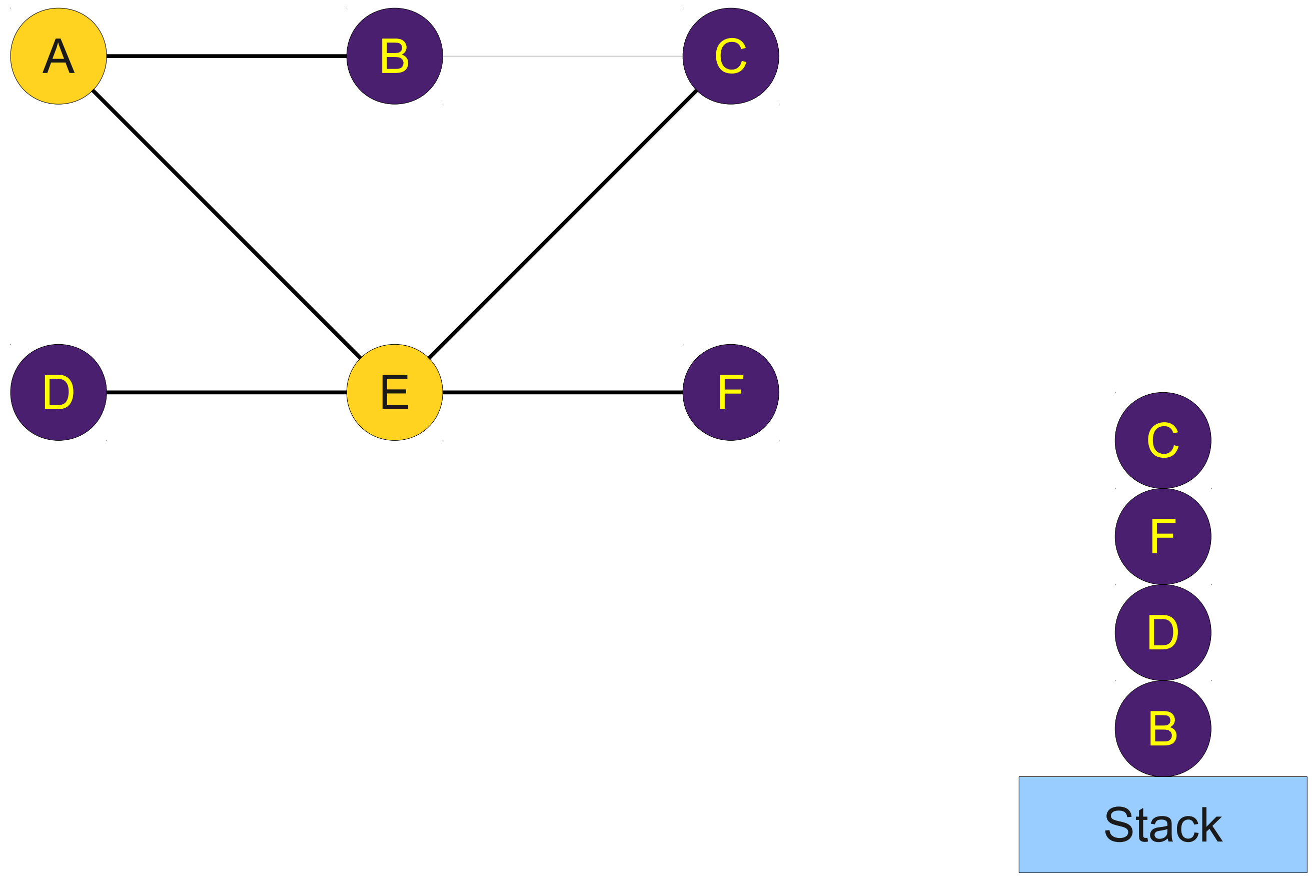
Depth-first search



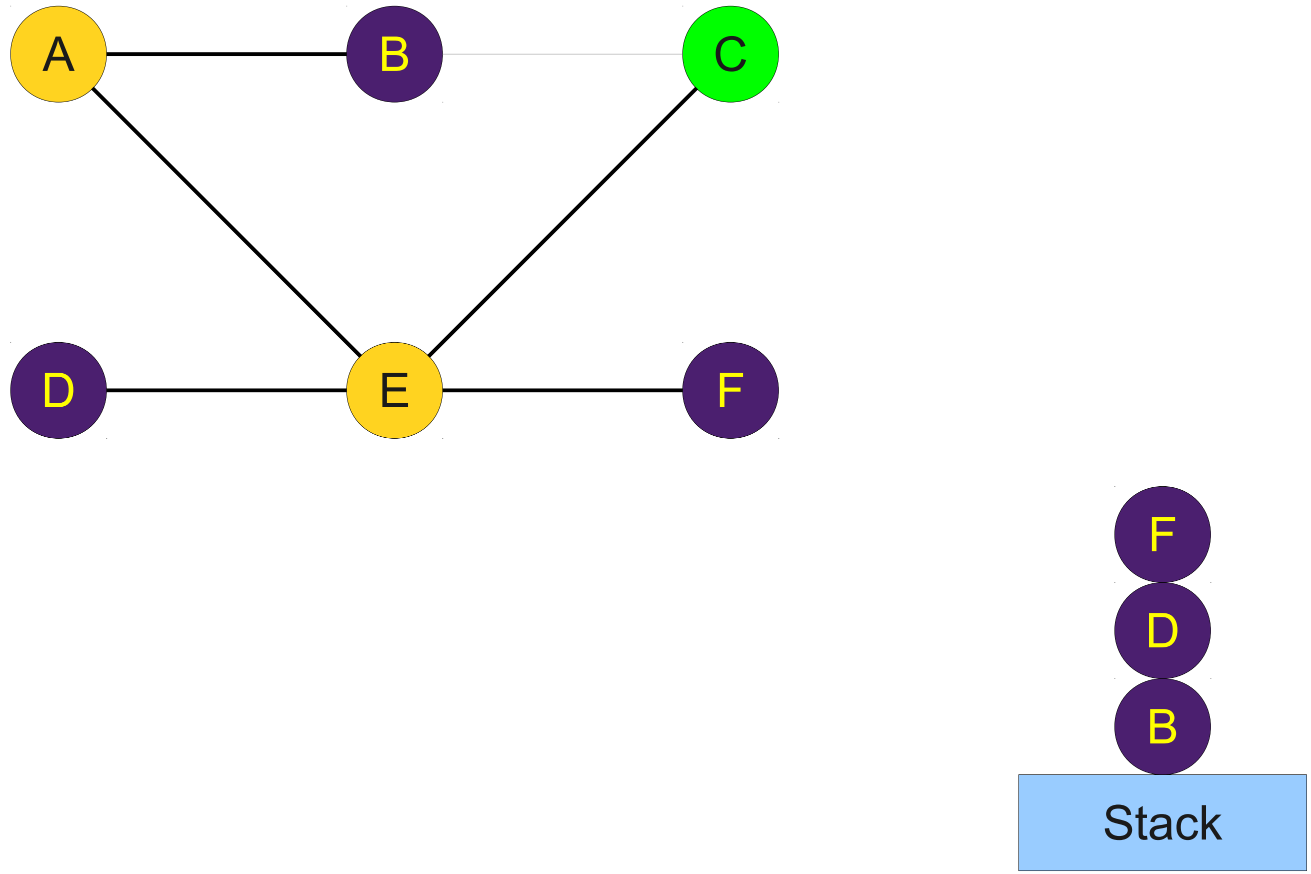
Depth-first search



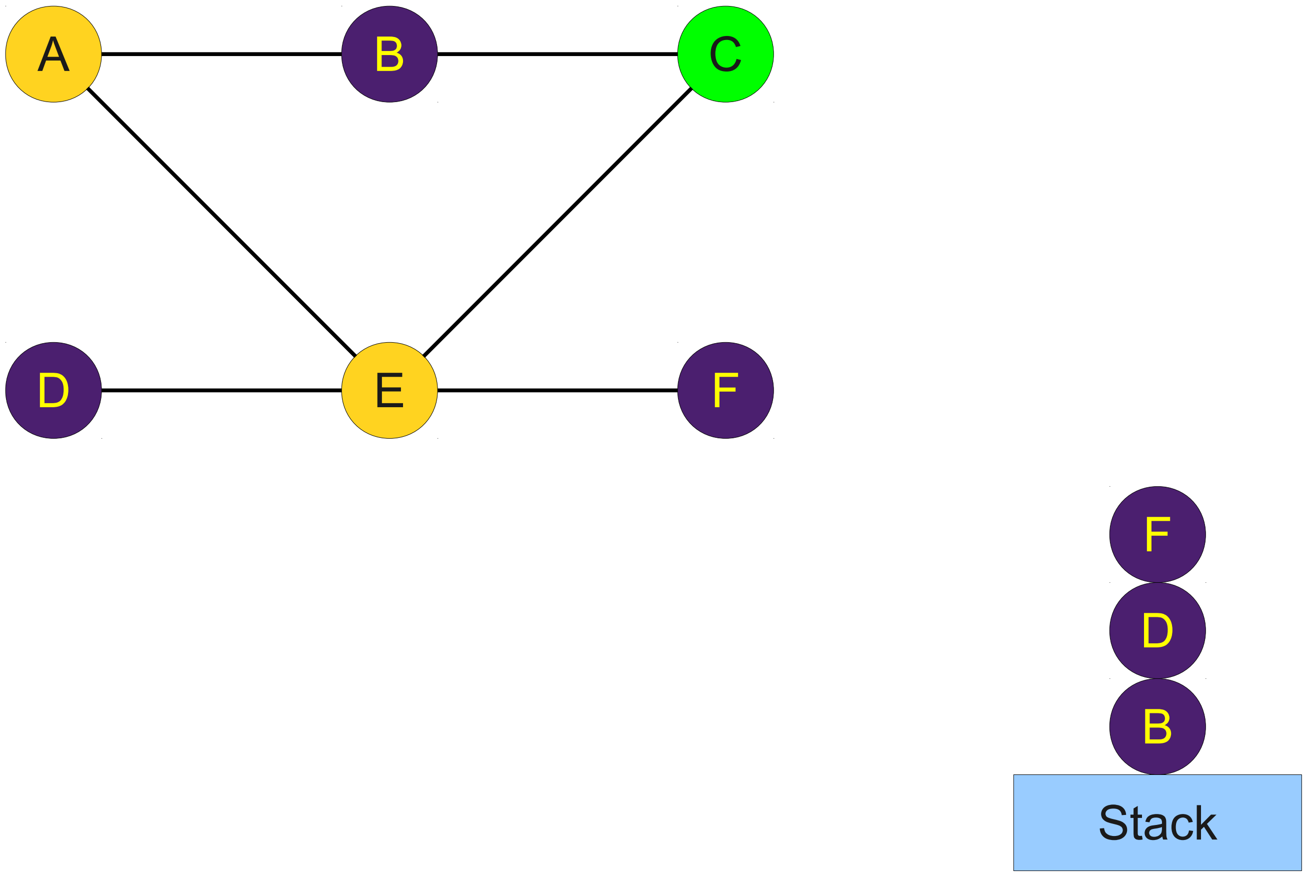
Depth-first search



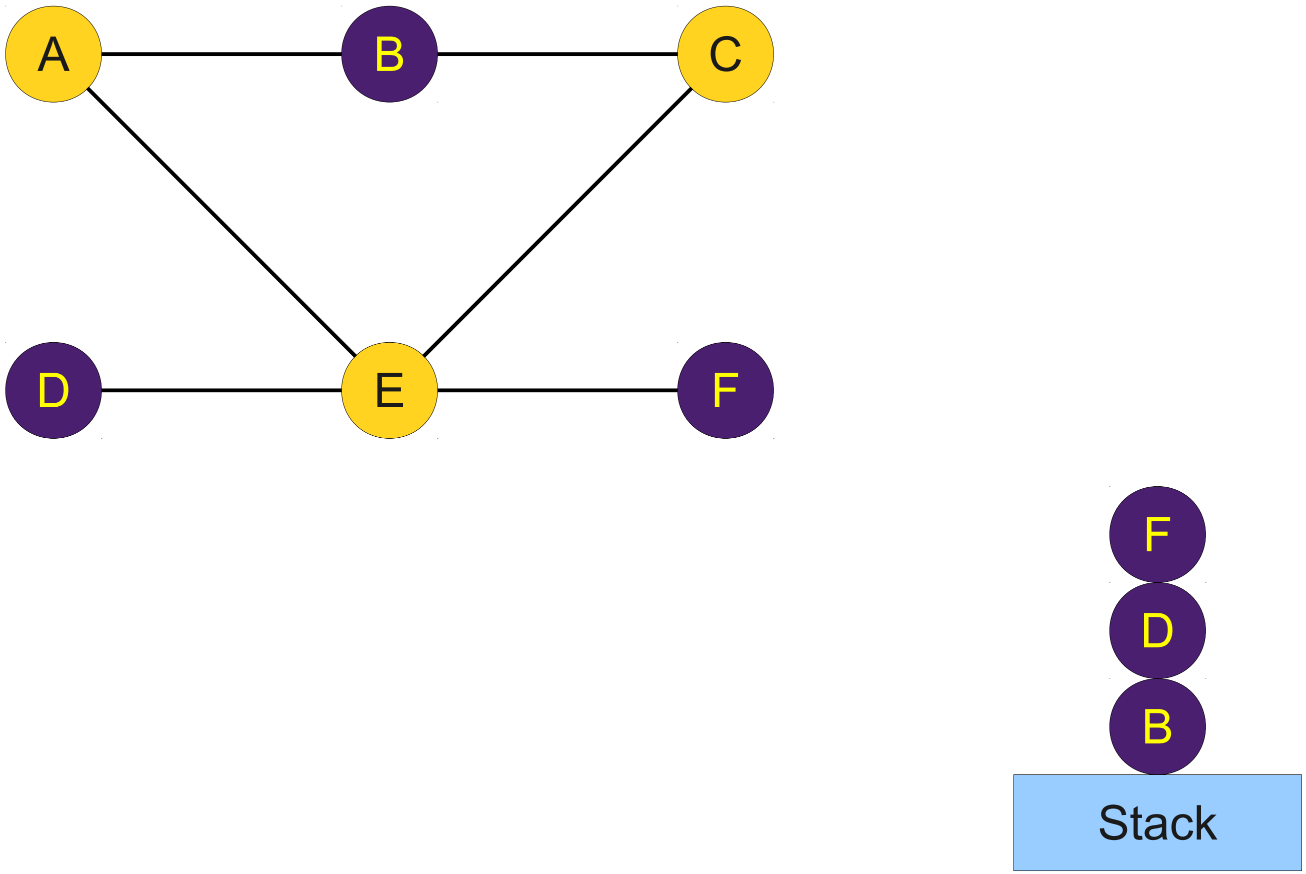
Depth-first search



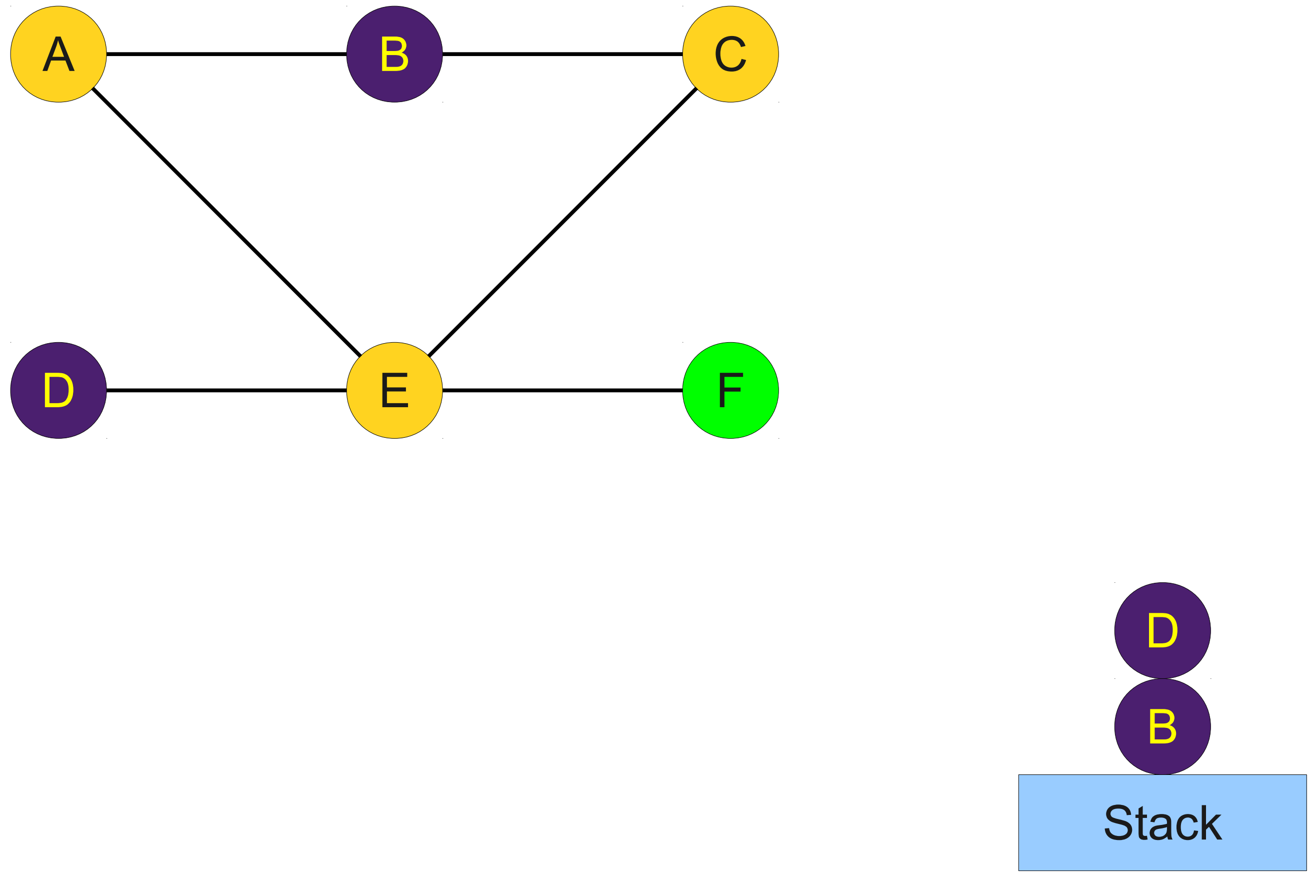
Depth-first search



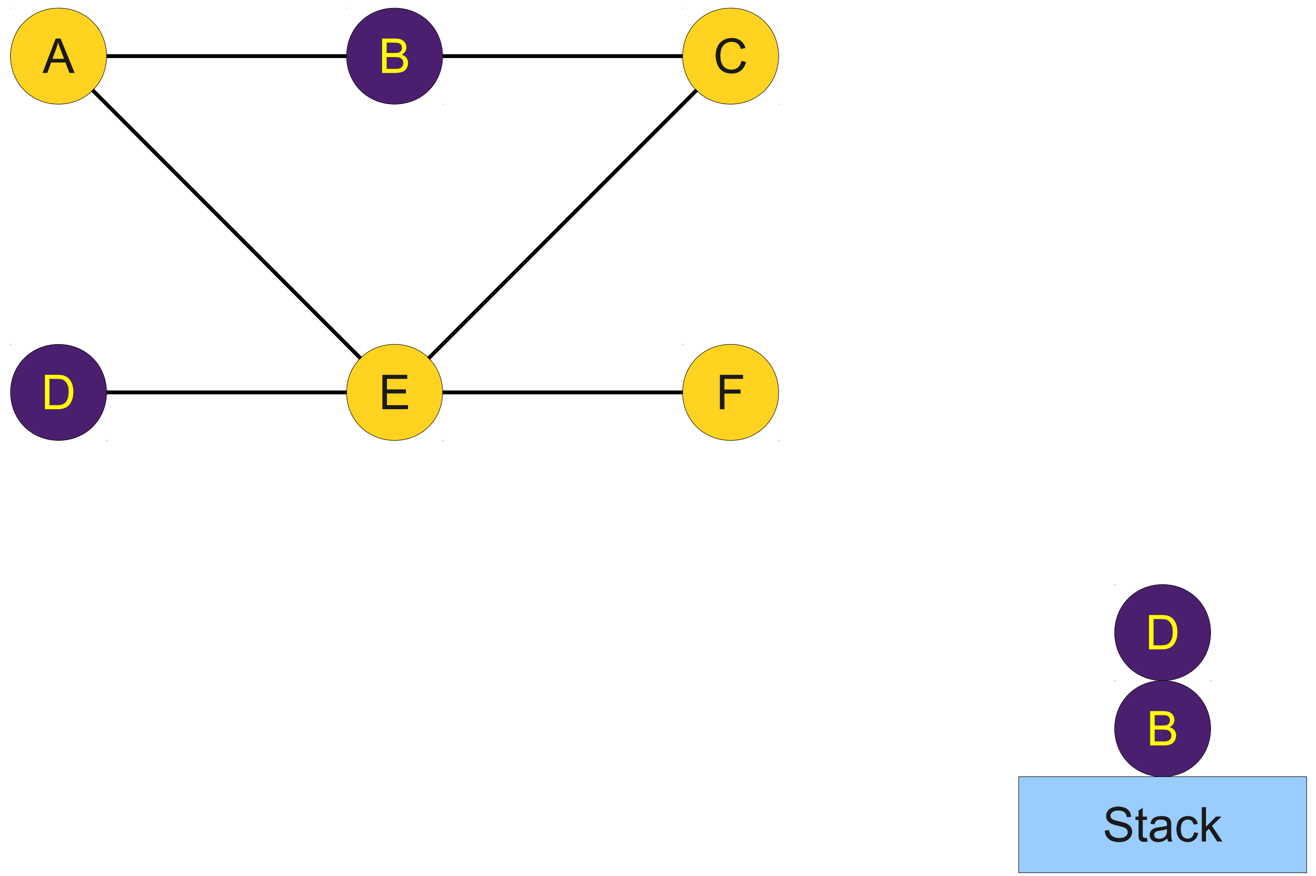
Depth-first search



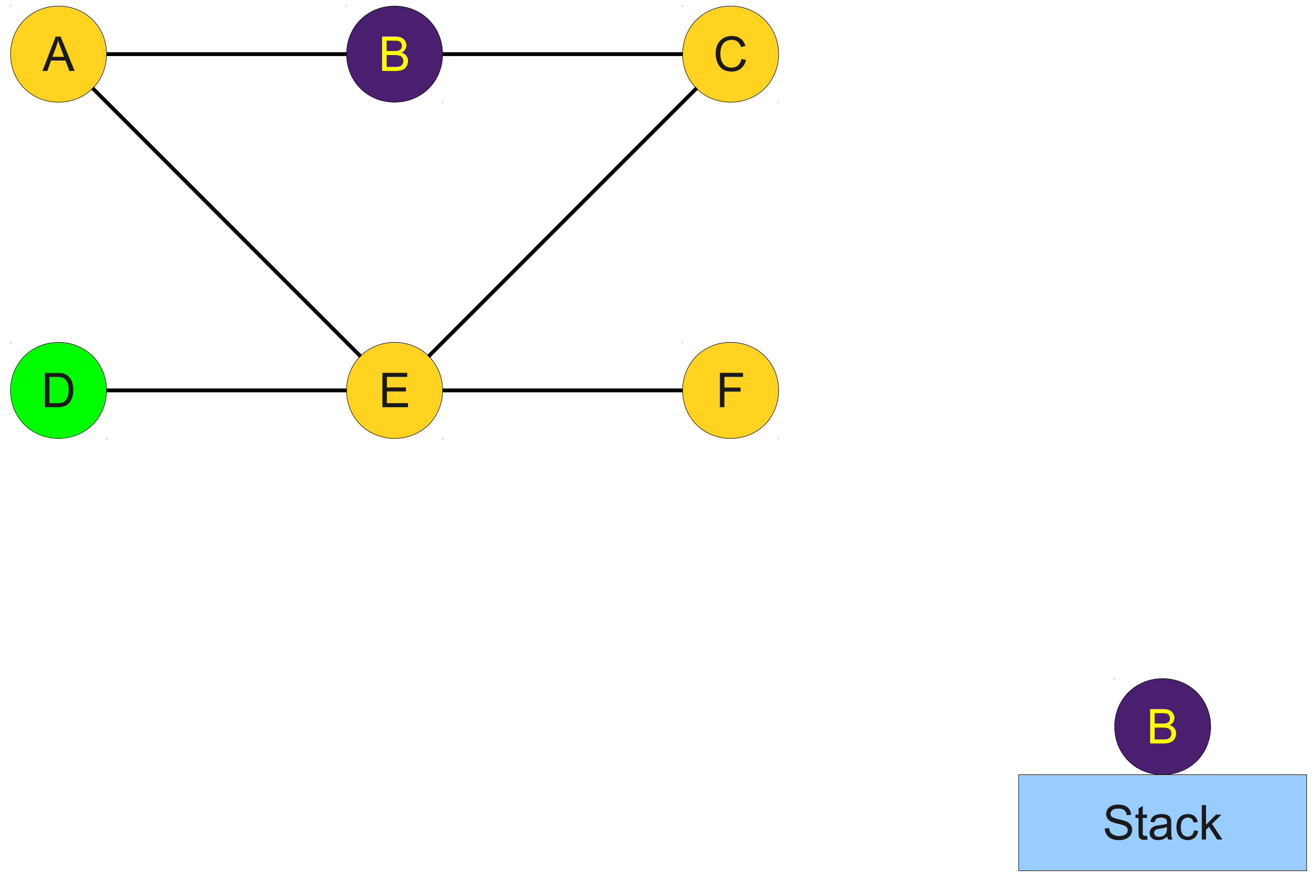
Depth-first search



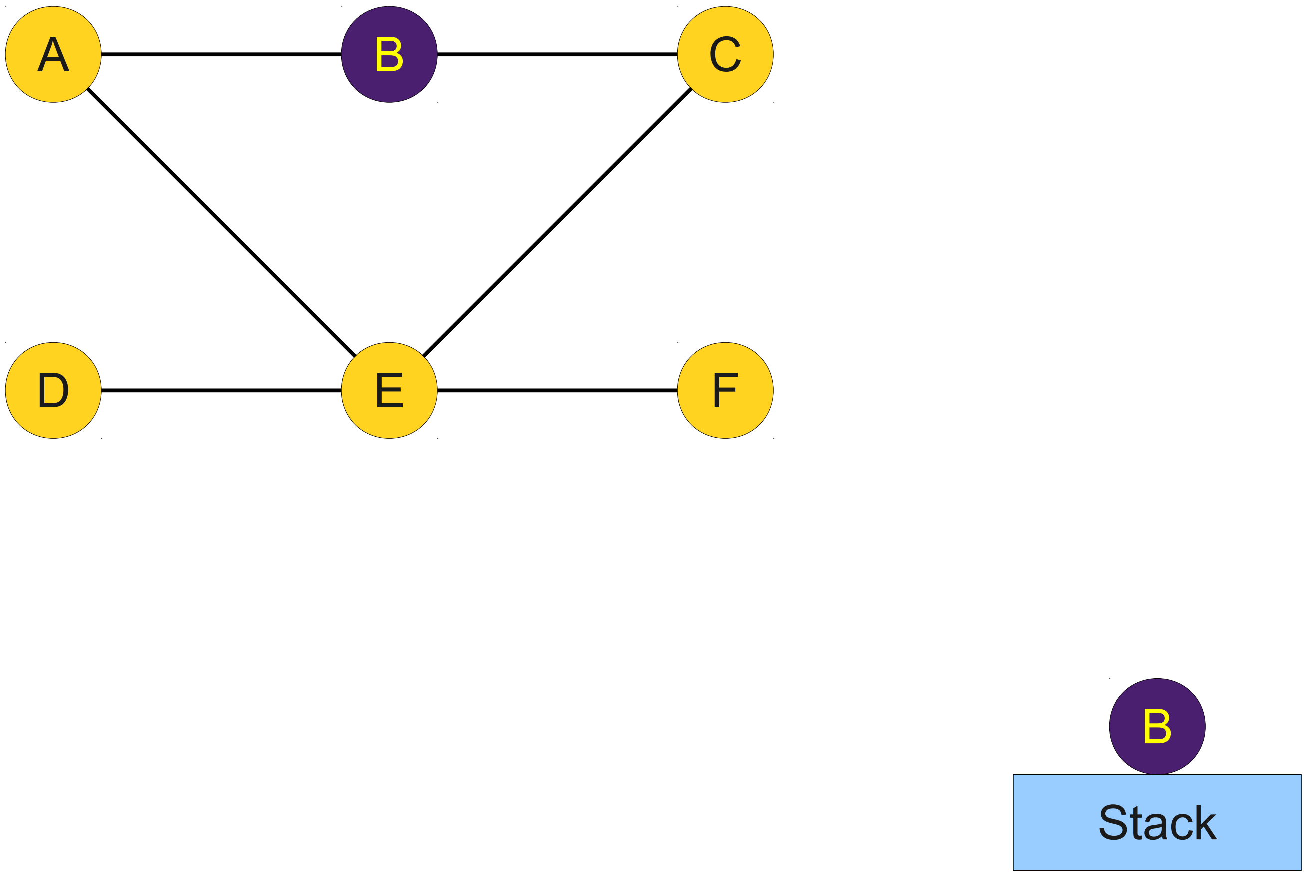
Depth-first search



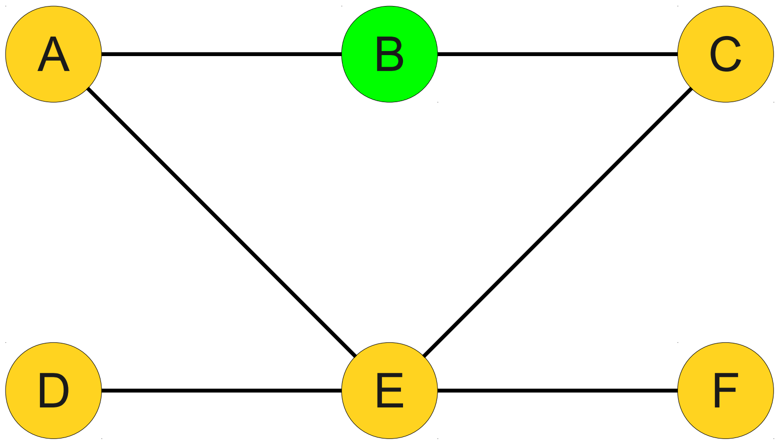
Depth-first search



Depth-first search

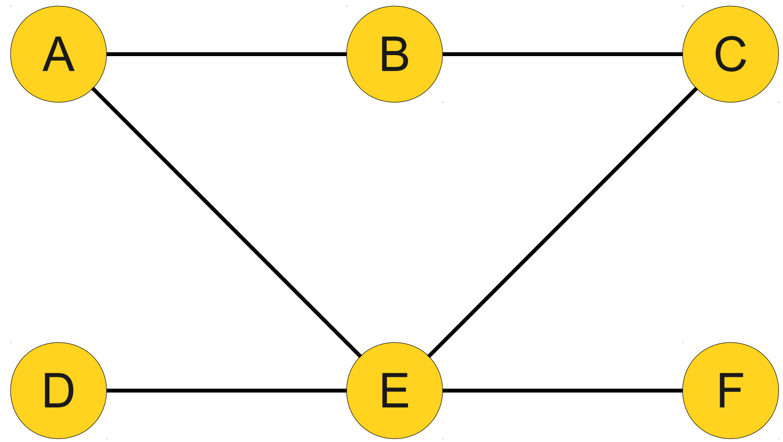


Depth-first search



Stack

Depth-first search

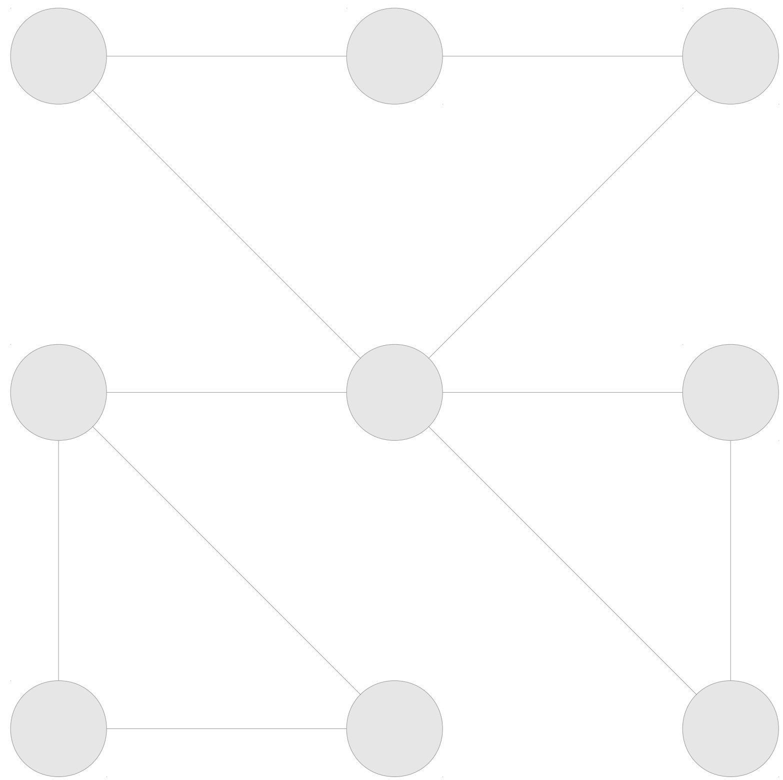


Stack

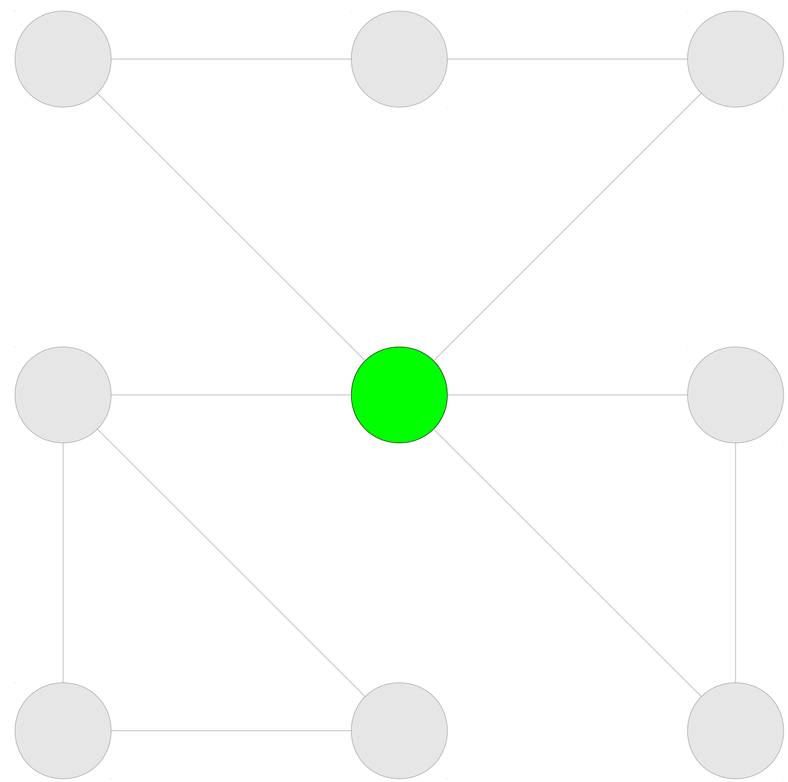
Iterative DFS

```
DFS(Node v, Set<Node> visited) {
   Create a Stack<Node> of nodes to visit;
   Add v to the stack;

   while (The stack is not empty) {
      Pop a node from the stack, let it be u;

      if (u has been visited) continue;
      Add u to the visited set;

      for (Node w connected to u)
         Push w onto the stack;
   }
}
```

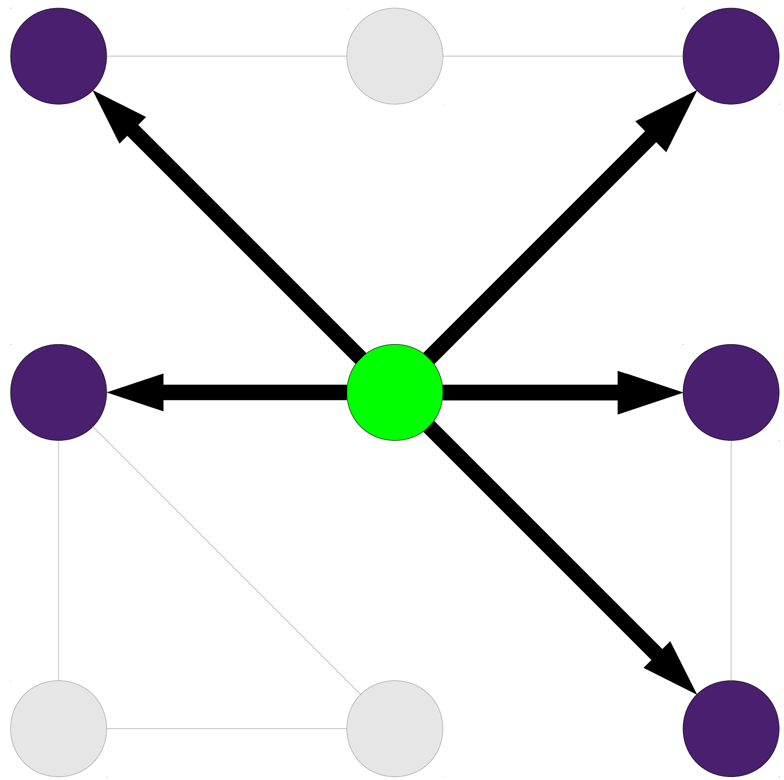
Graph Search Trees



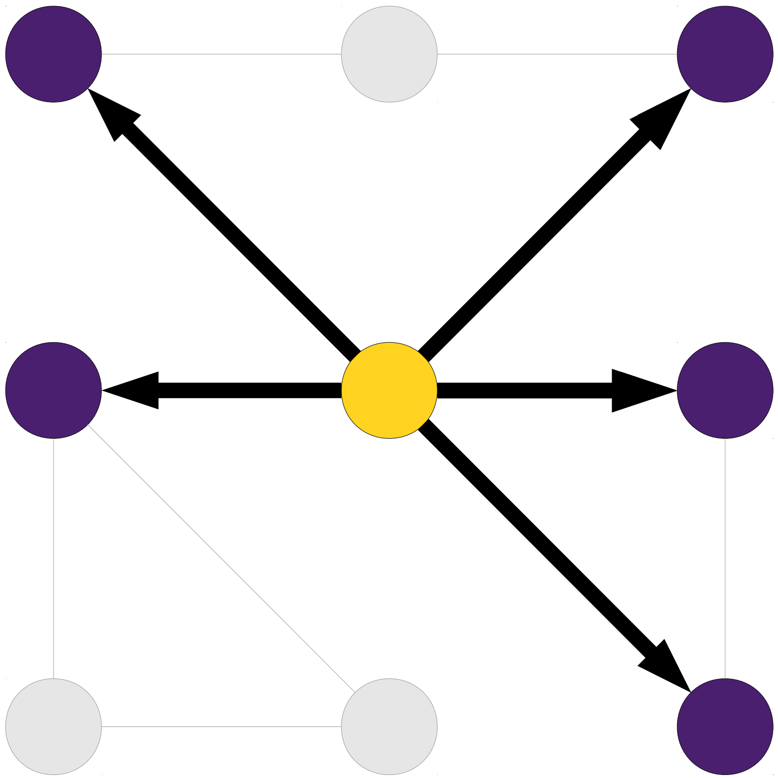
Graph Search Trees



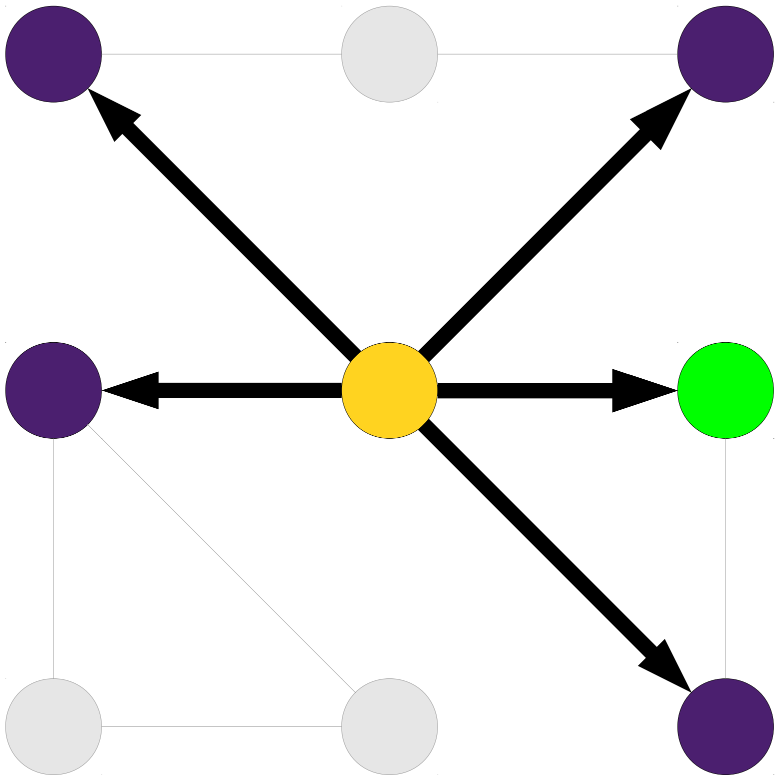
Graph Search Trees



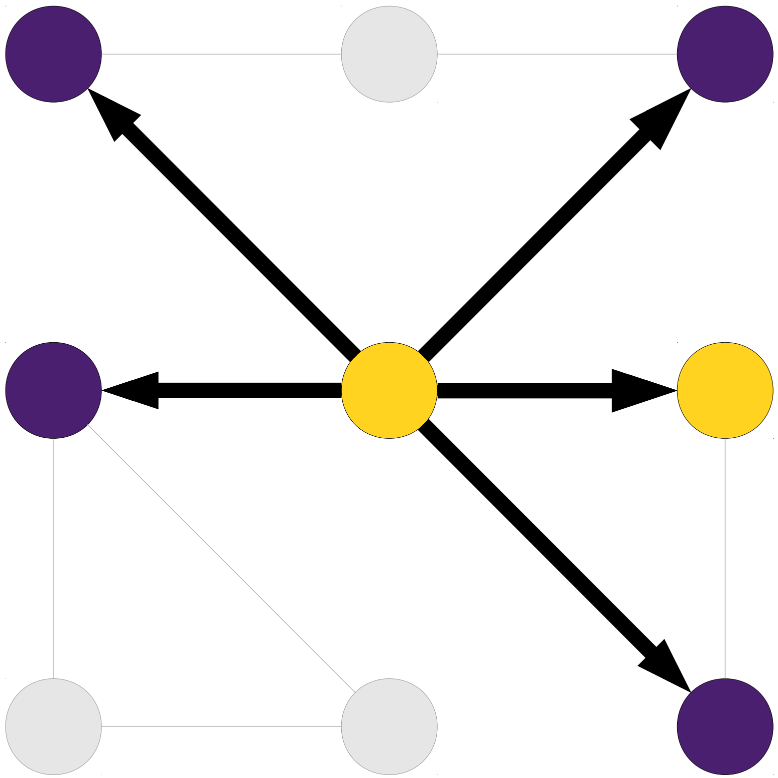
Graph Search Trees



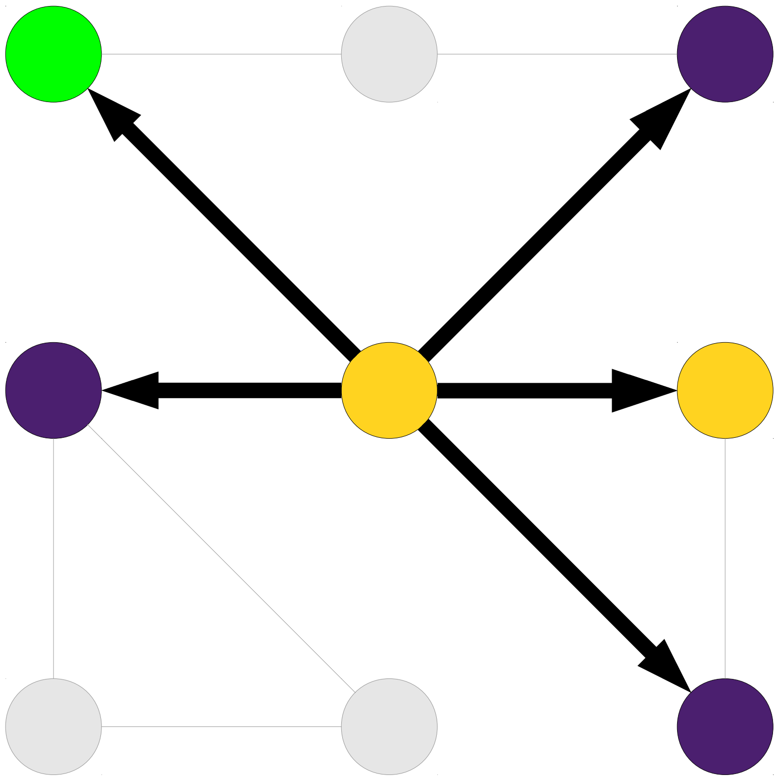
Graph Search Trees



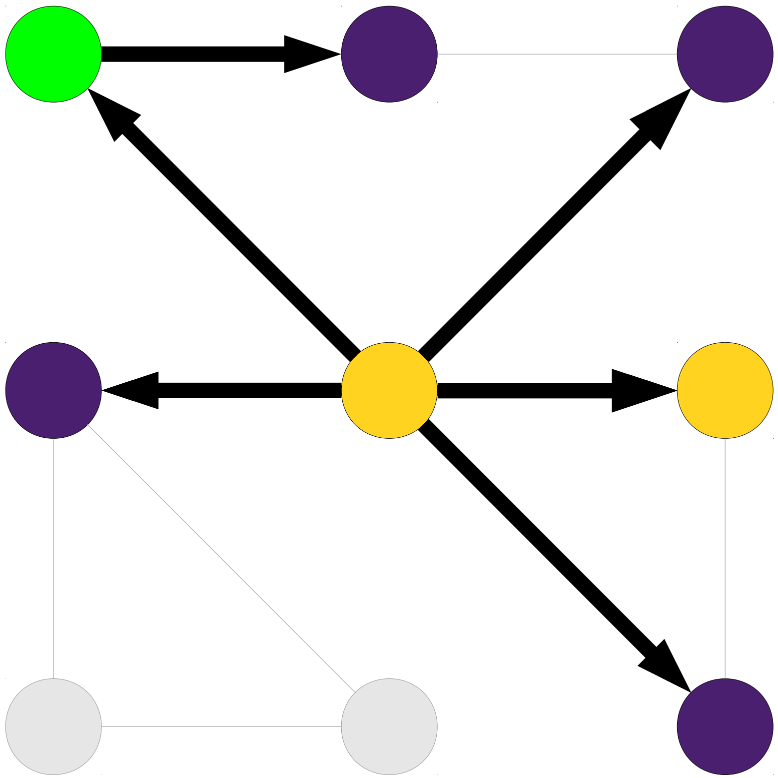
Graph Search Trees



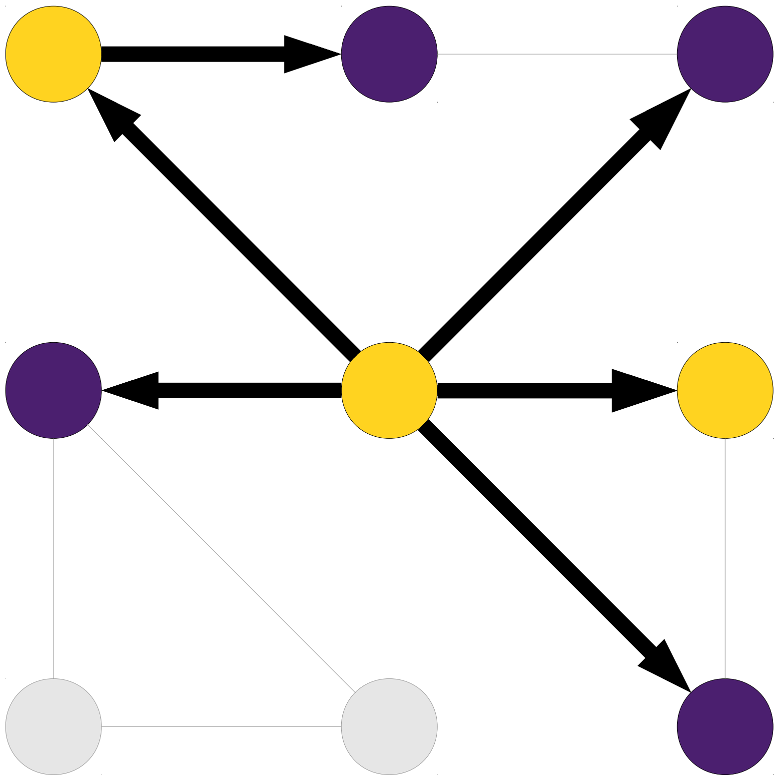
Graph Search Trees



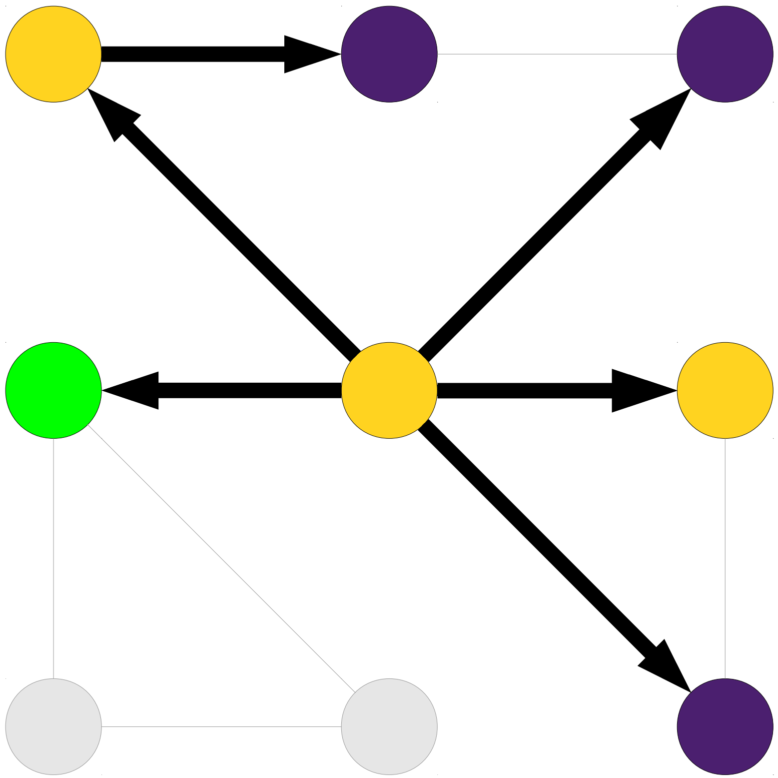
Graph Search Trees



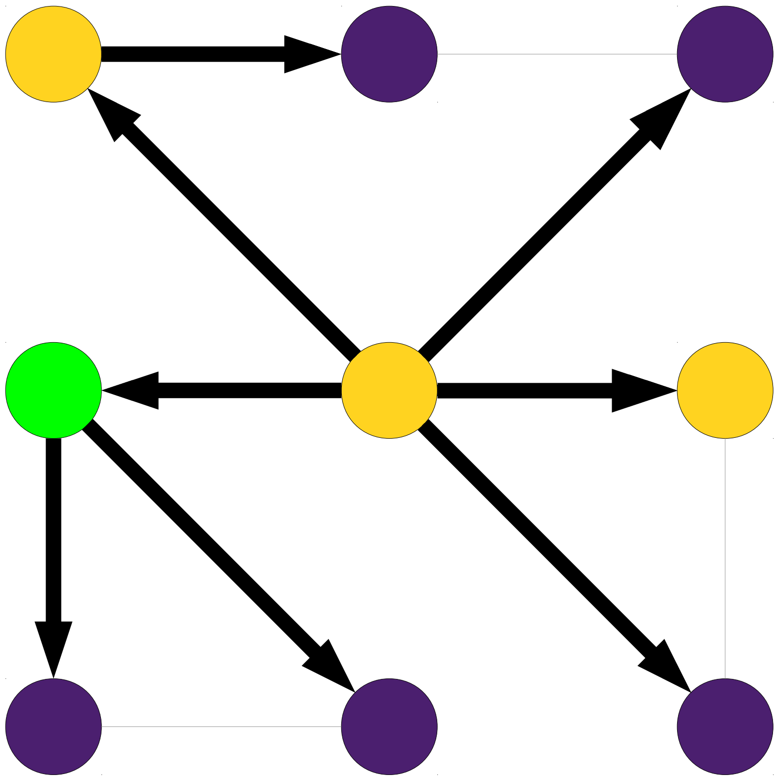
Graph Search Trees



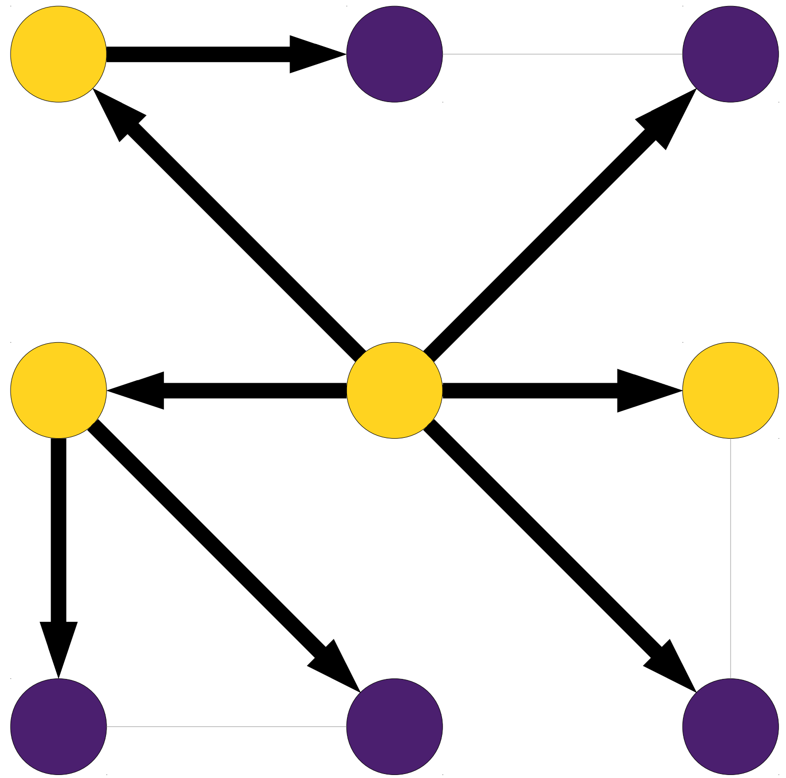
Graph Search Trees



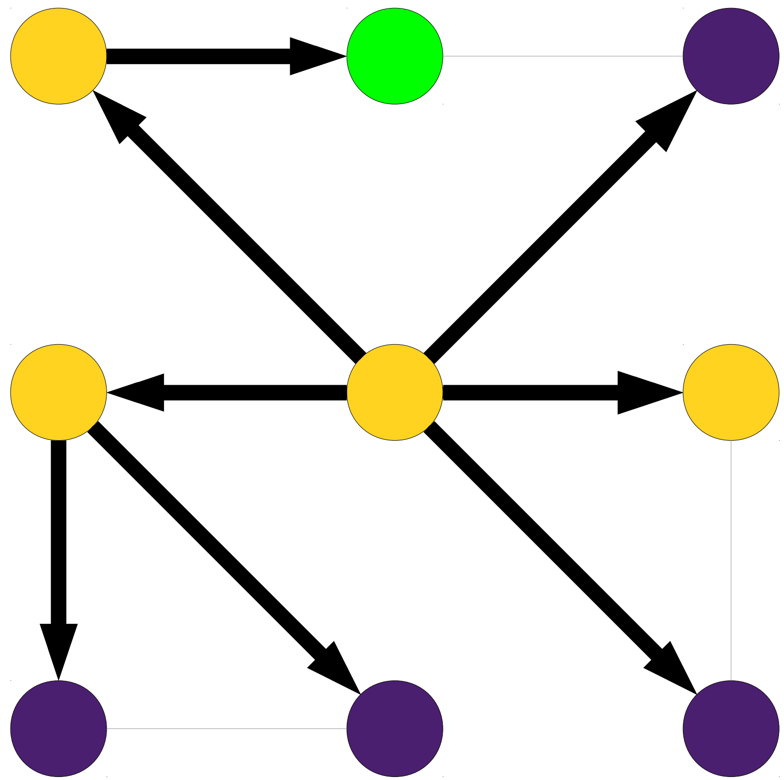
Graph Search Trees



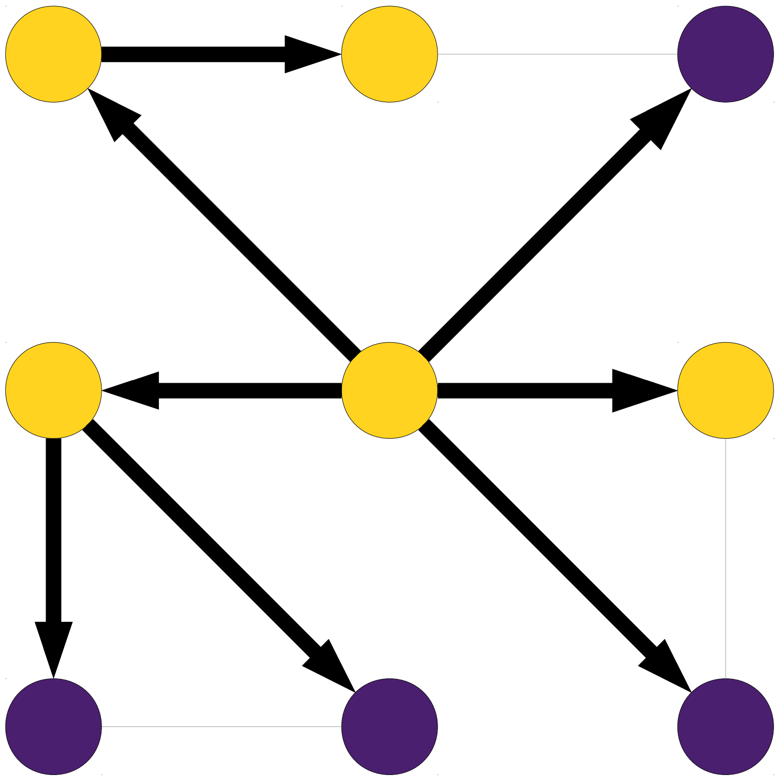
Graph Search Trees



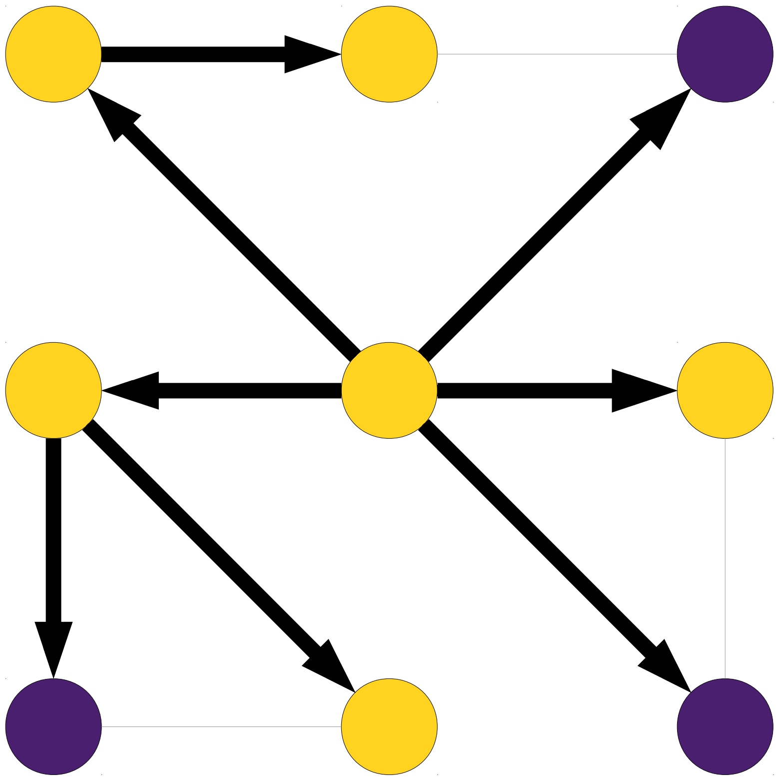
Graph Search Trees



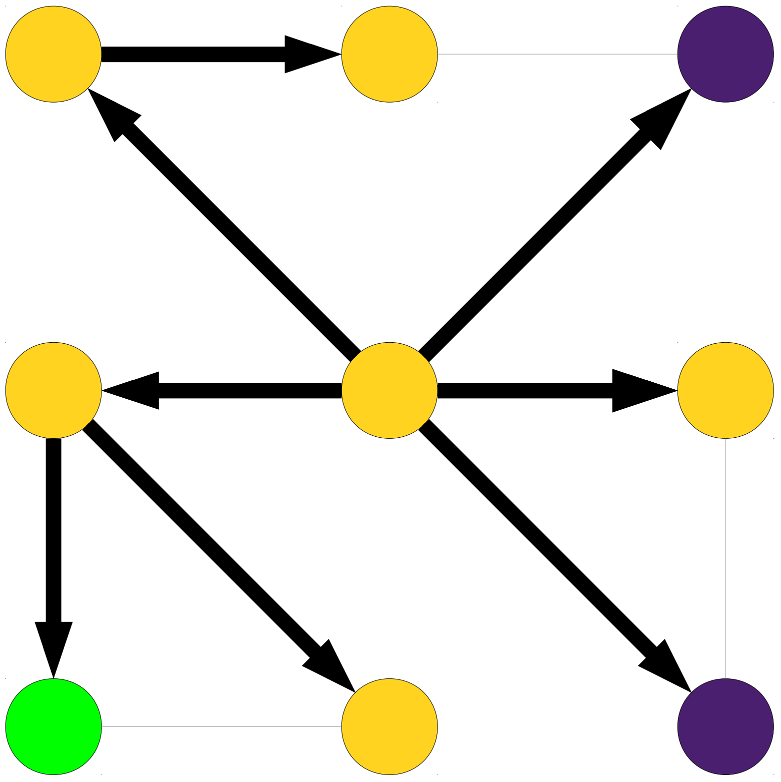
Graph Search Trees



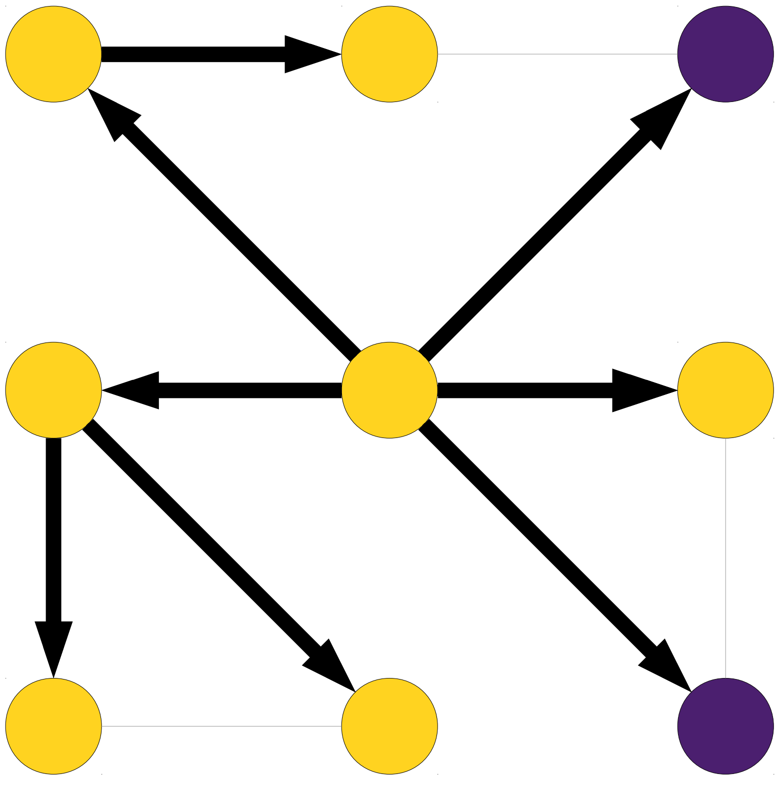
Graph Search Trees



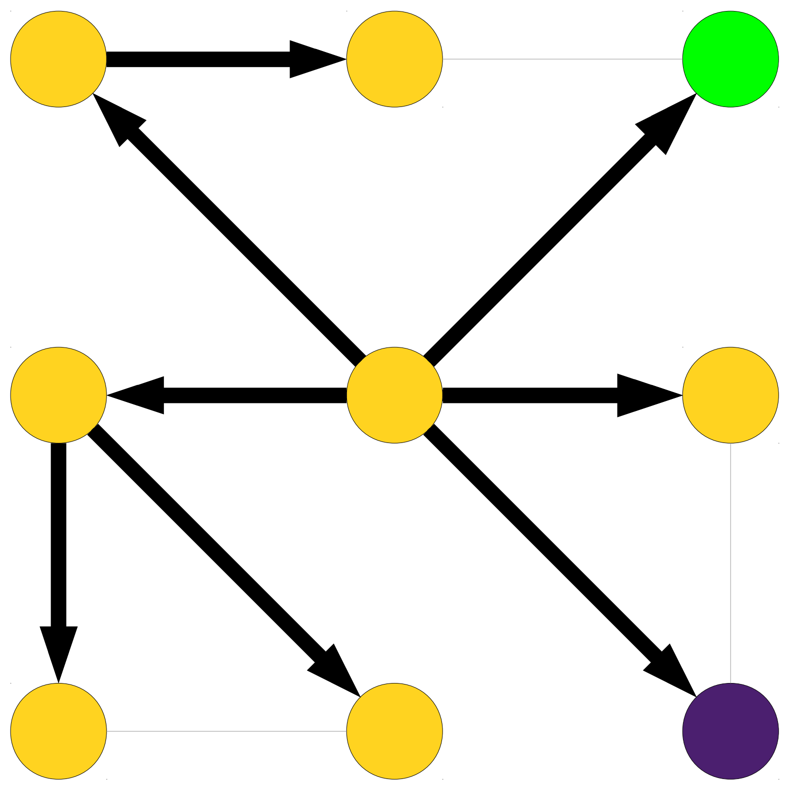
Graph Search Trees



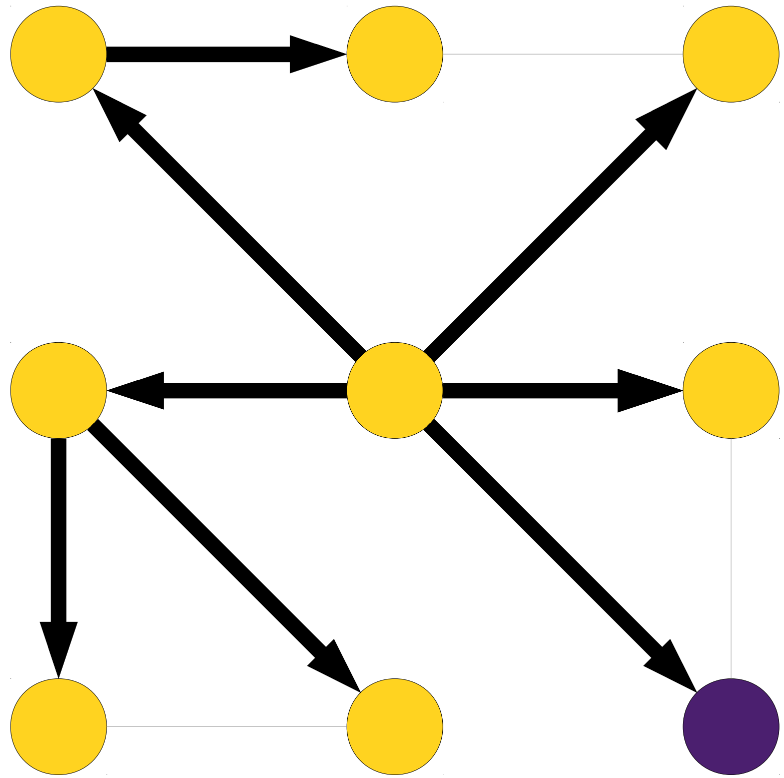
Graph Search Trees



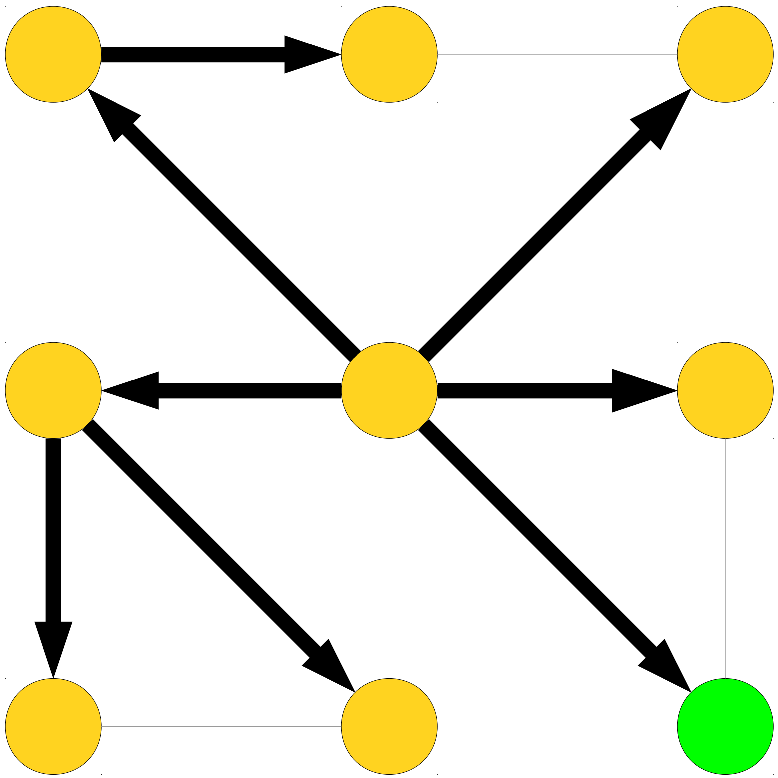
Graph Search Trees



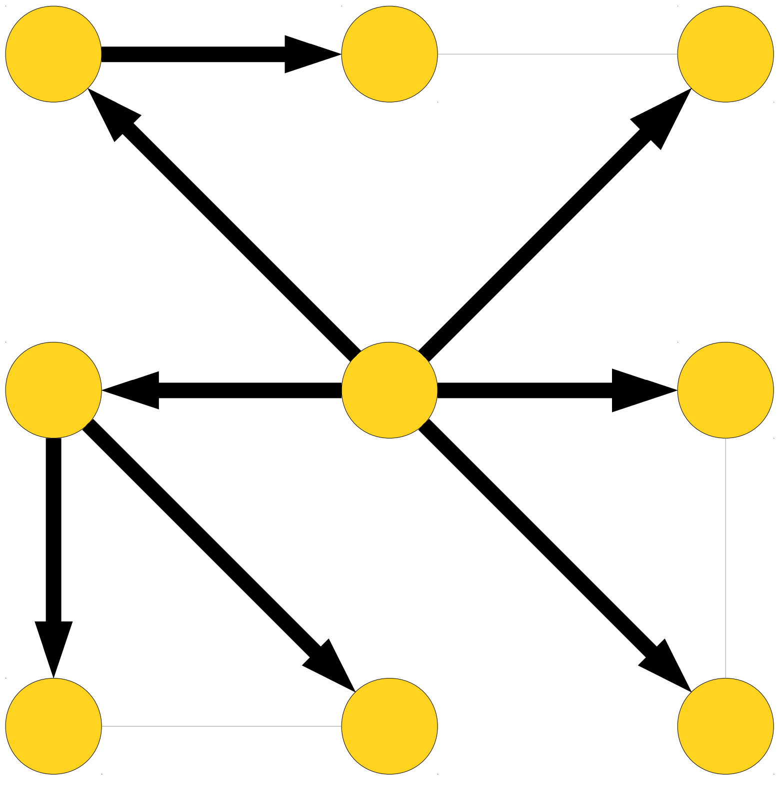
Graph Search Trees



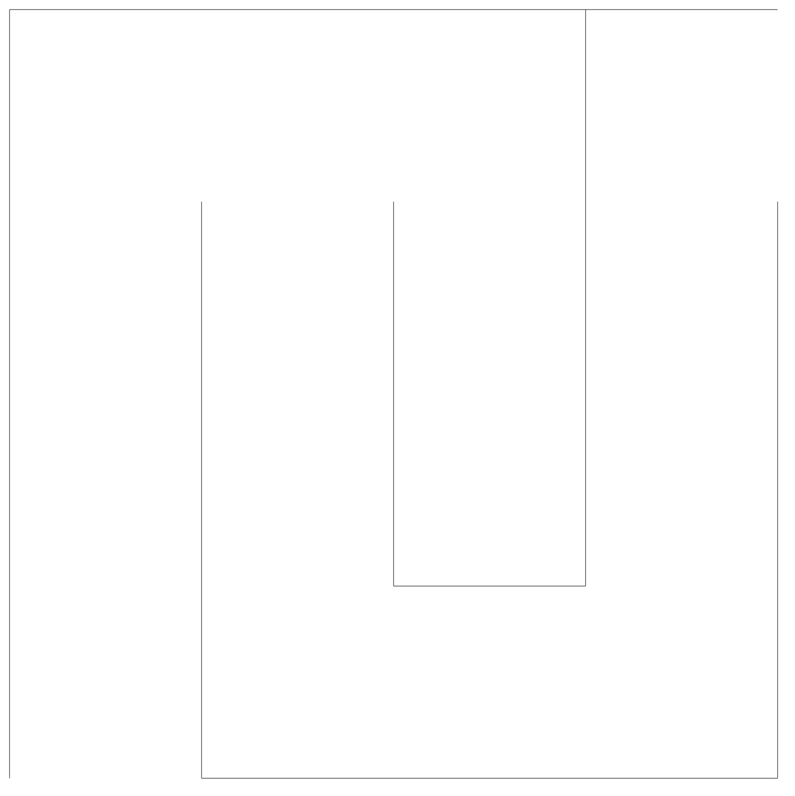
Graph Search Trees



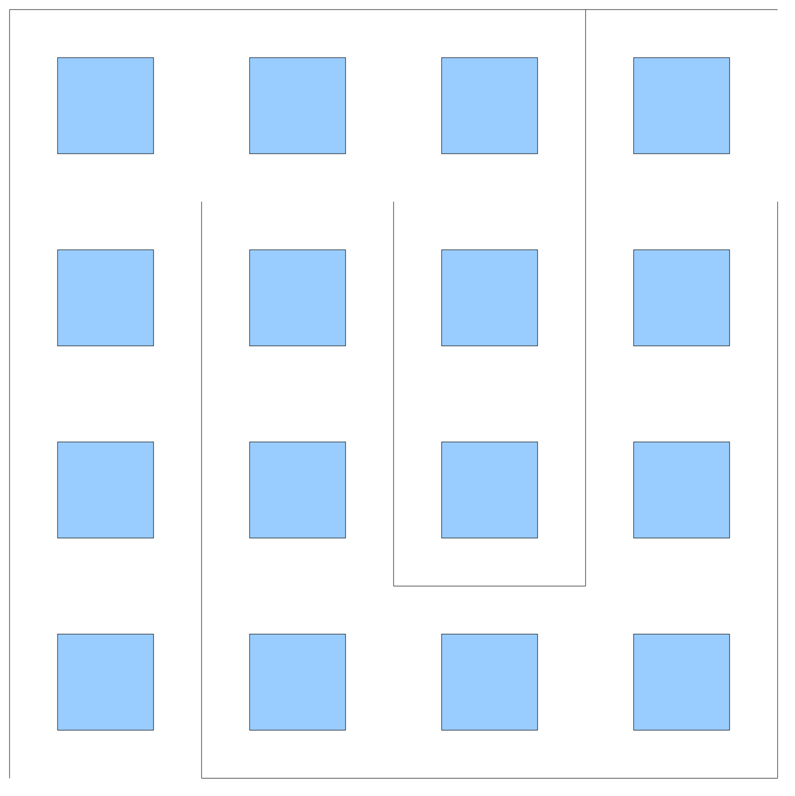
Graph Search Trees



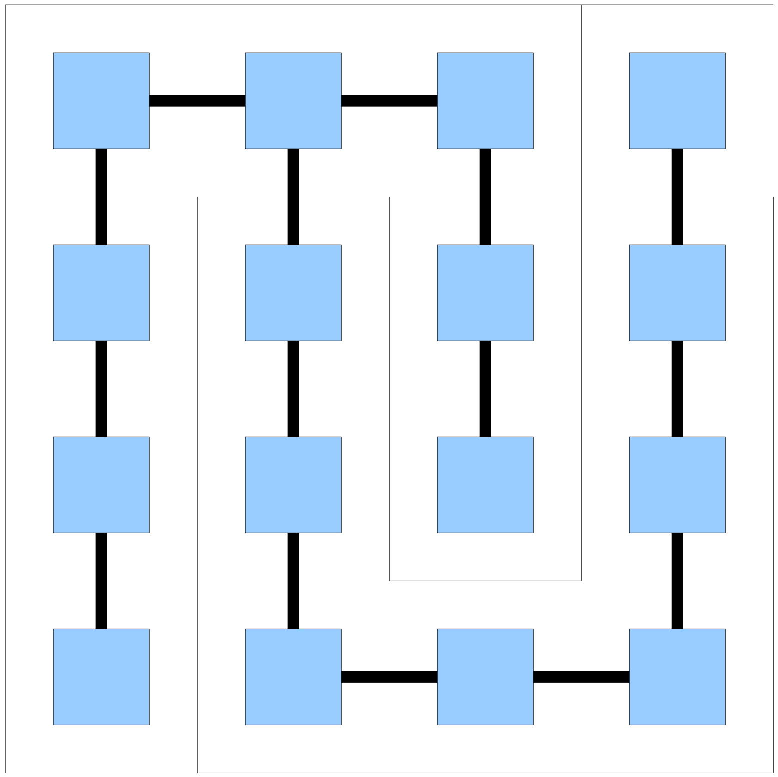
Mazes as Graphs



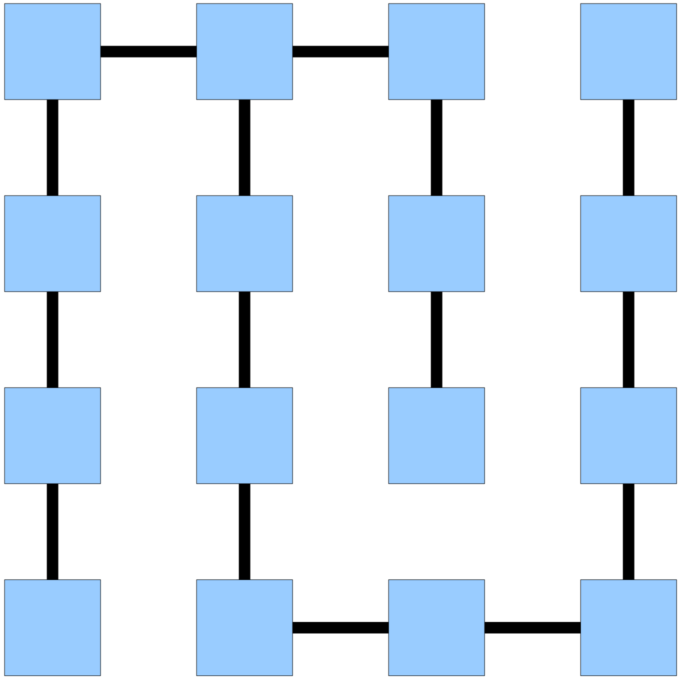
Mazes as Graphs



Mazes as Graphs



Mazes as Graphs

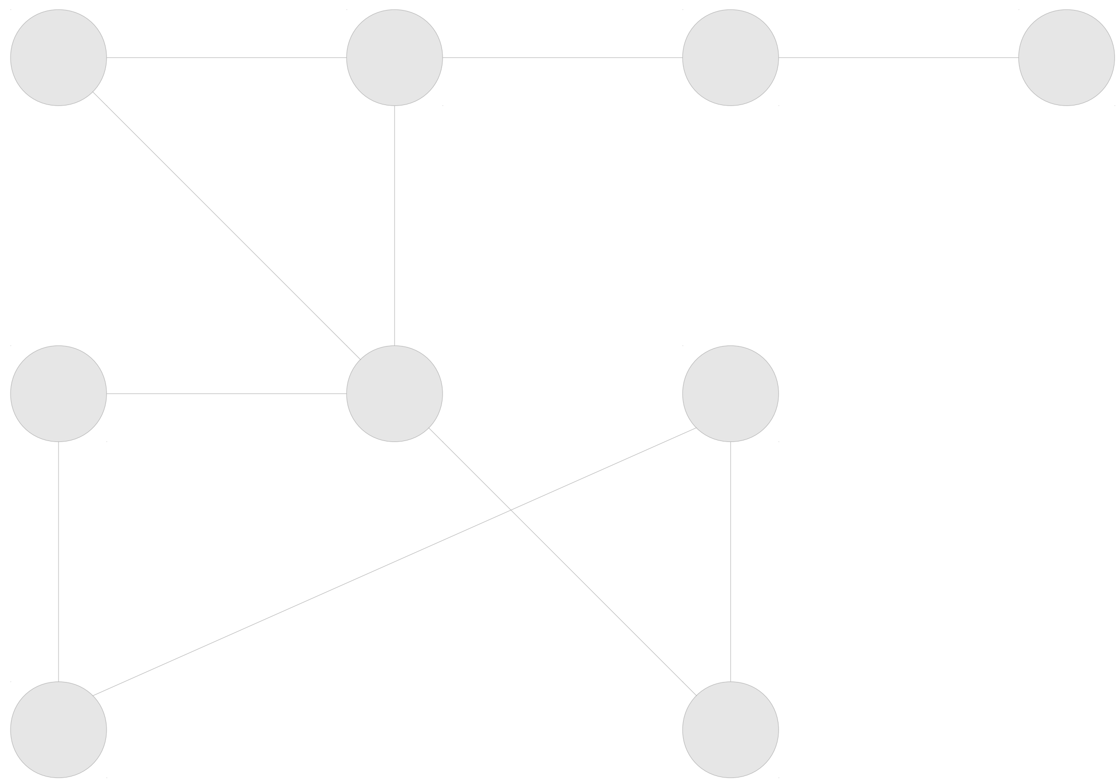


Creating a Maze with DFS

- Create a **grid graph** of the appropriate size.
- Starting at any node, run a depth-first search, adding the arcs to the stack in random order.
- The resulting DFS tree is a maze with one solution.

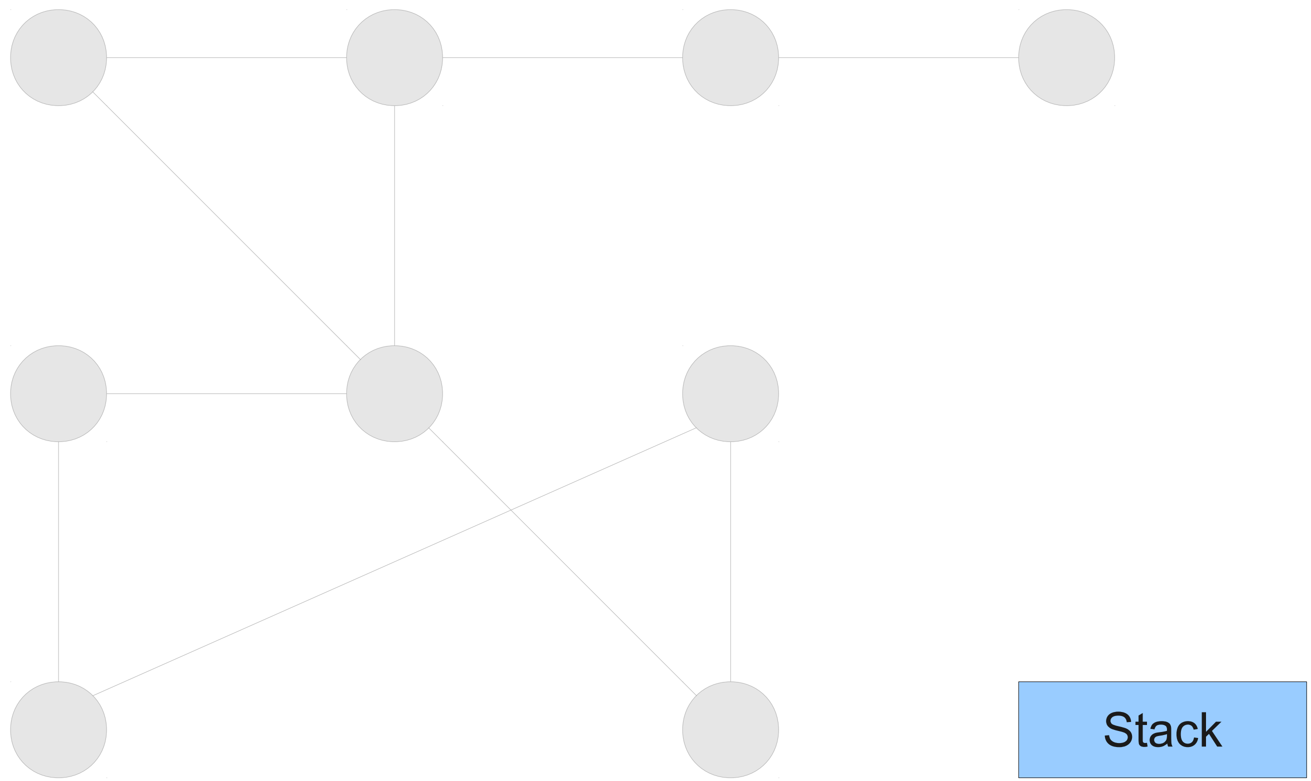
Problems with DFS

- Useful when trying to explore everything.
- Not good at finding specific nodes.



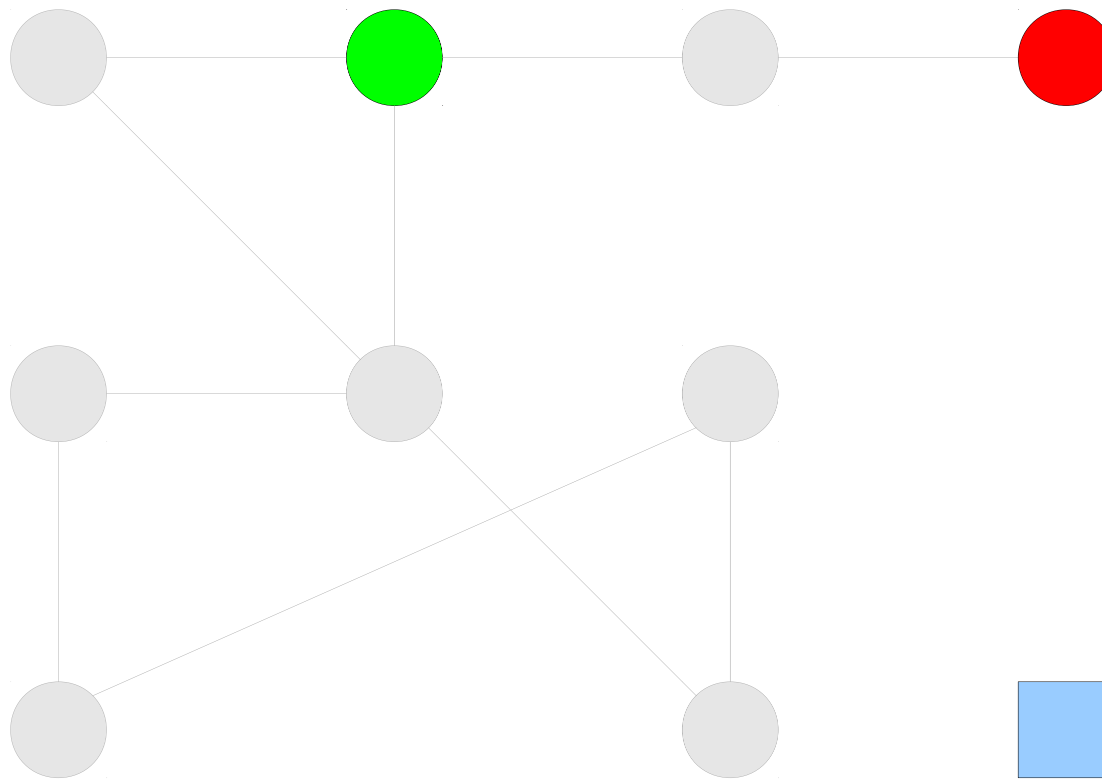
Problems with DFS

- Useful when trying to explore everything.
- Not good at finding specific nodes.



Problems with DFS

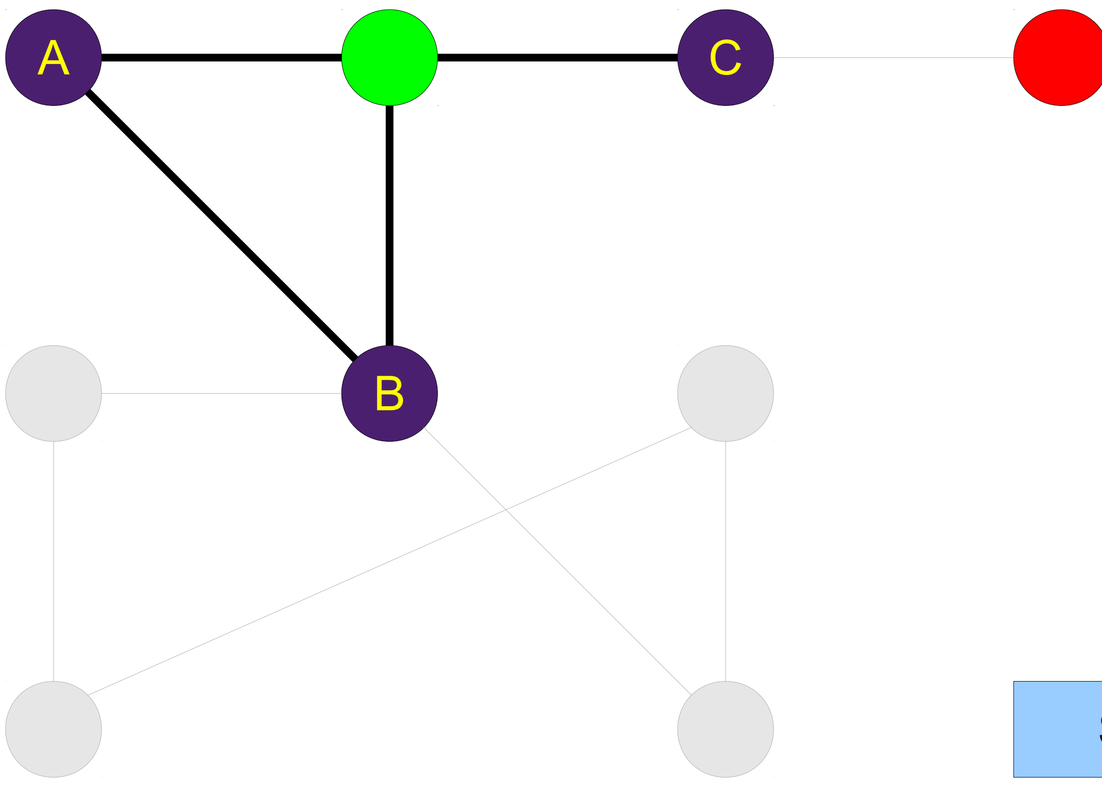
- Useful when trying to explore everything.
- Not good at finding specific nodes.

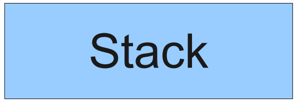


Stack

Problems with DFS

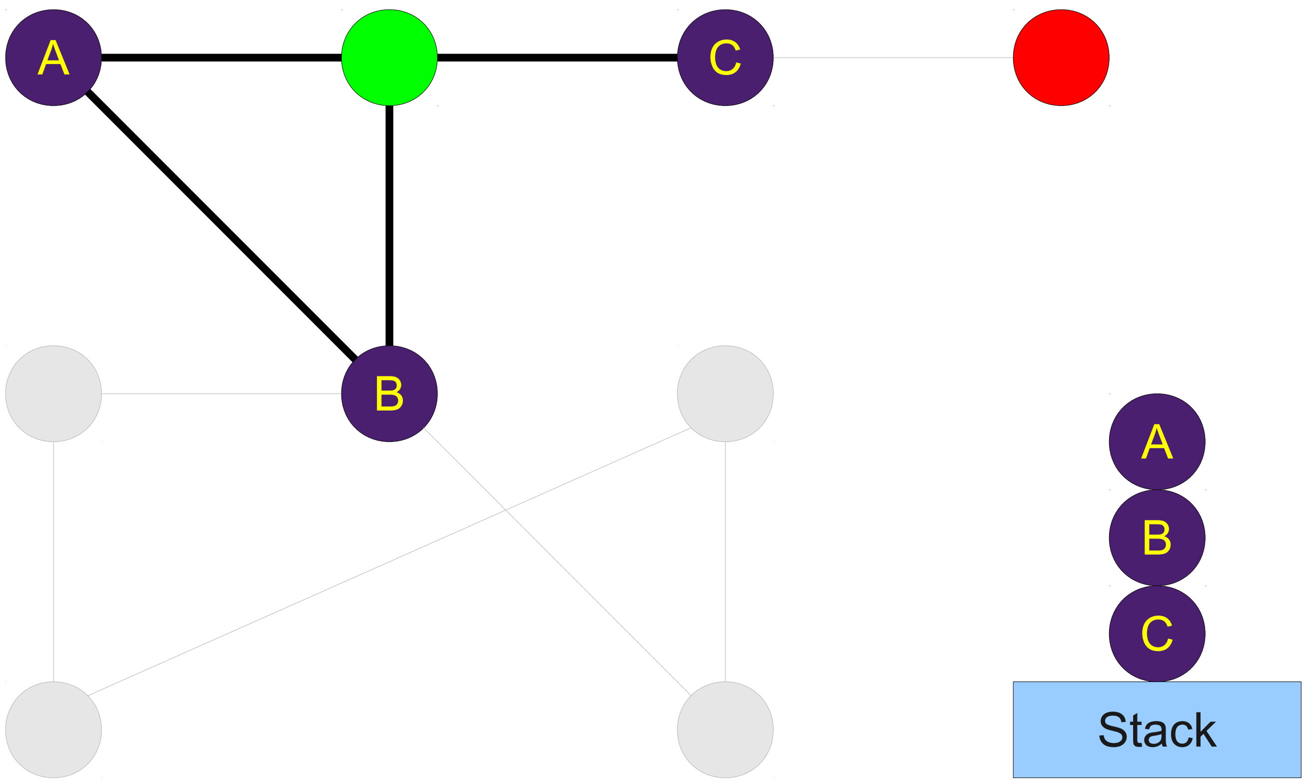
- Useful when trying to explore everything.
- Not good at finding specific nodes.





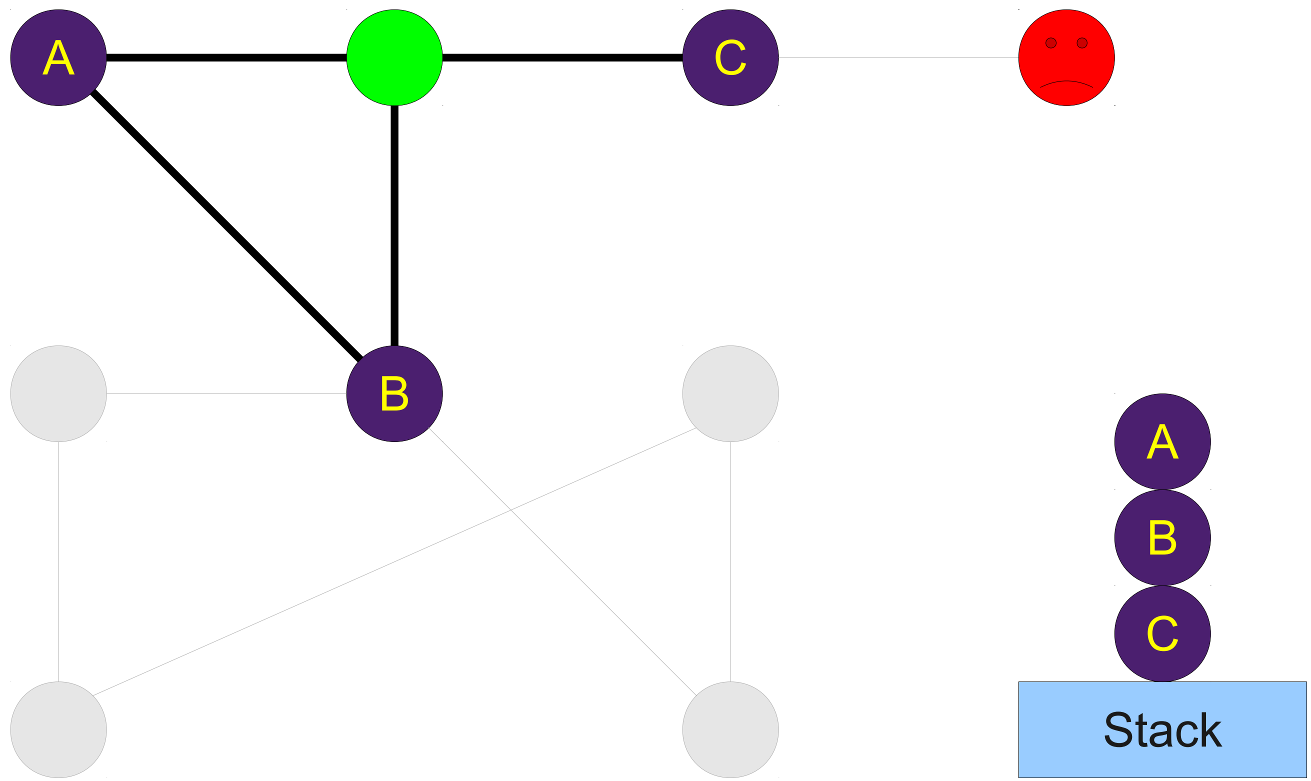
Problems with DFS

- Useful when trying to explore everything.
- Not good at finding specific nodes.



Problems with DFS

- Useful when trying to explore everything.
- Not good at finding specific nodes.

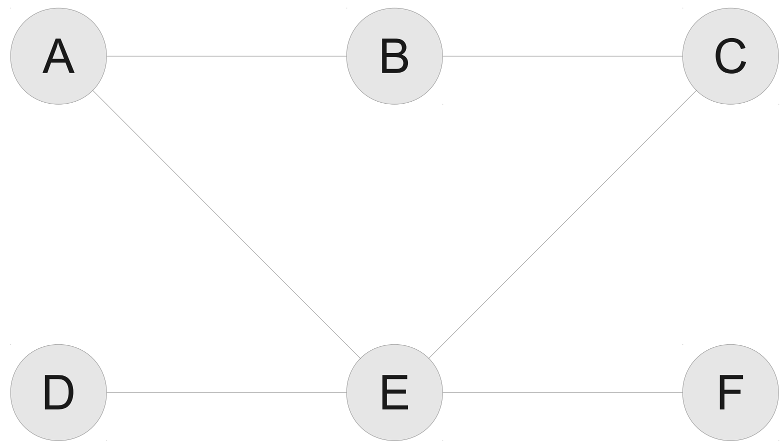


Breadth-First Search

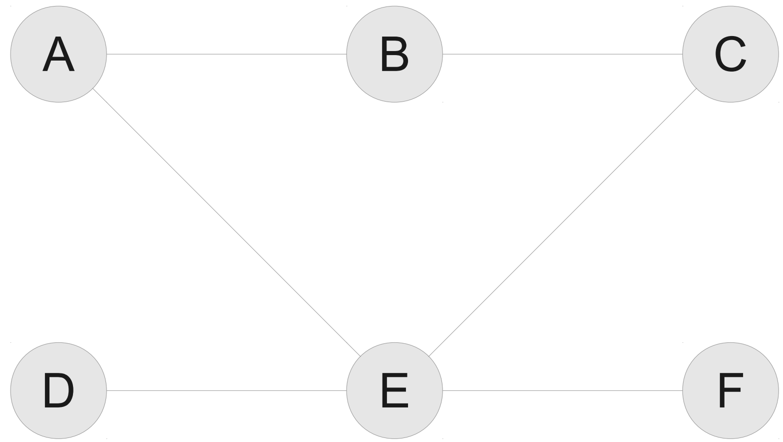
Breadth-First Search

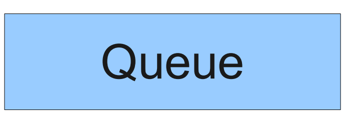
- Specialization of the general search algorithm where nodes to visit are put into a *queue*.
- Explores nodes one hop away, then two hops away, etc.
- Finds path with fewest edges from start node to all other nodes.

Breadth-first search

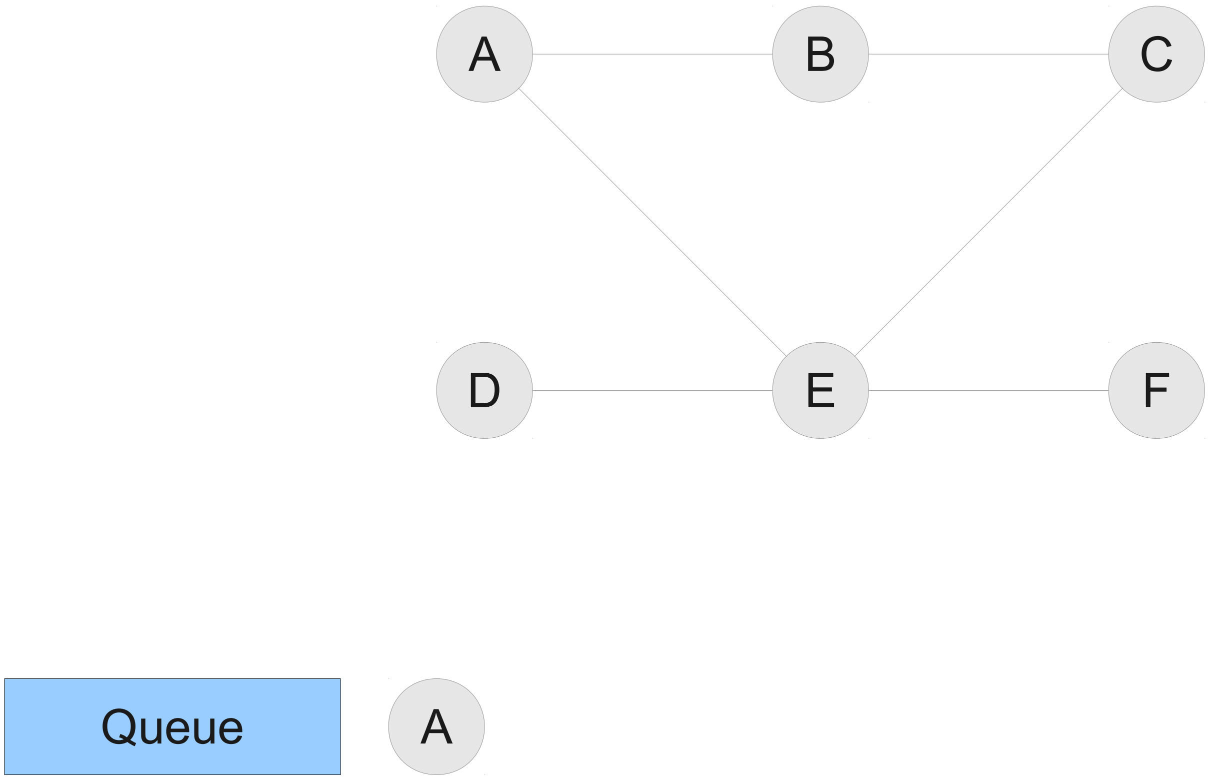


Breadth-first search

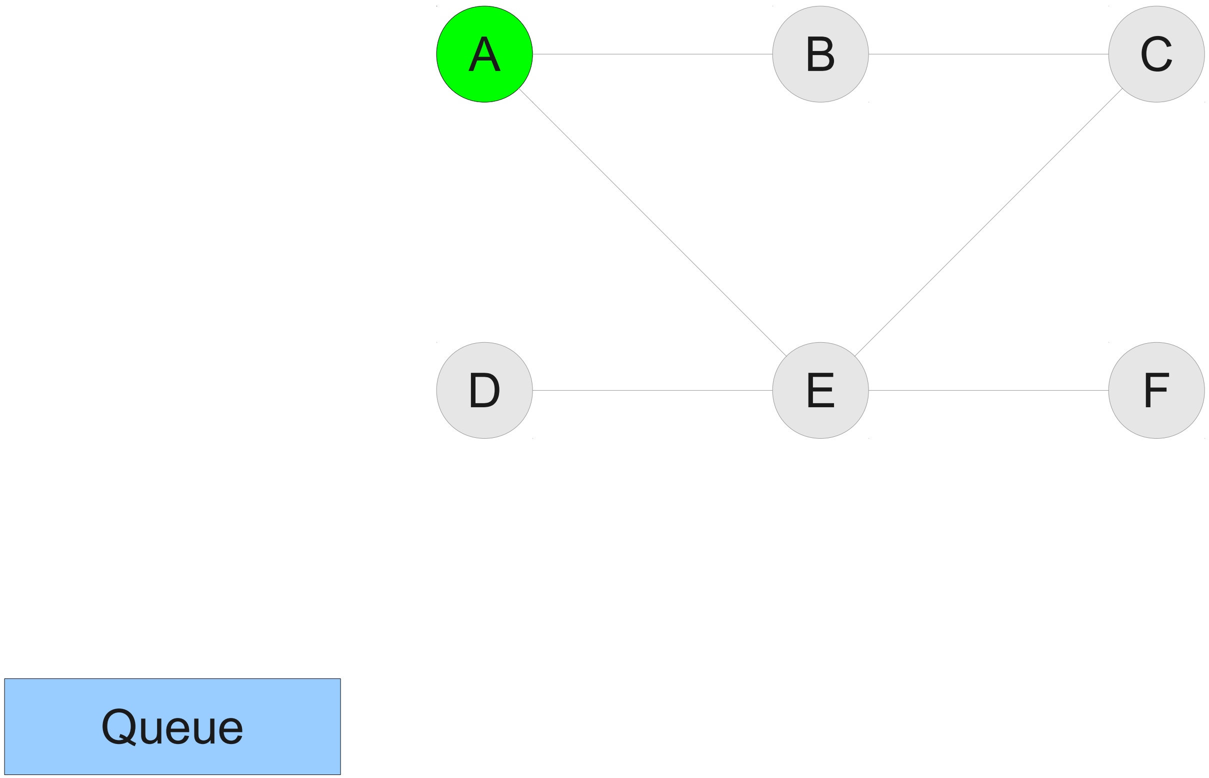




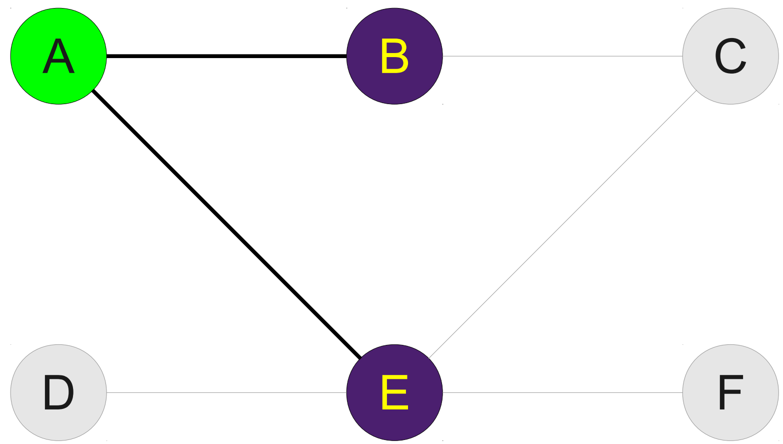
Breadth-first search

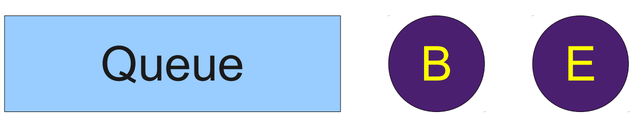


Breadth-first search

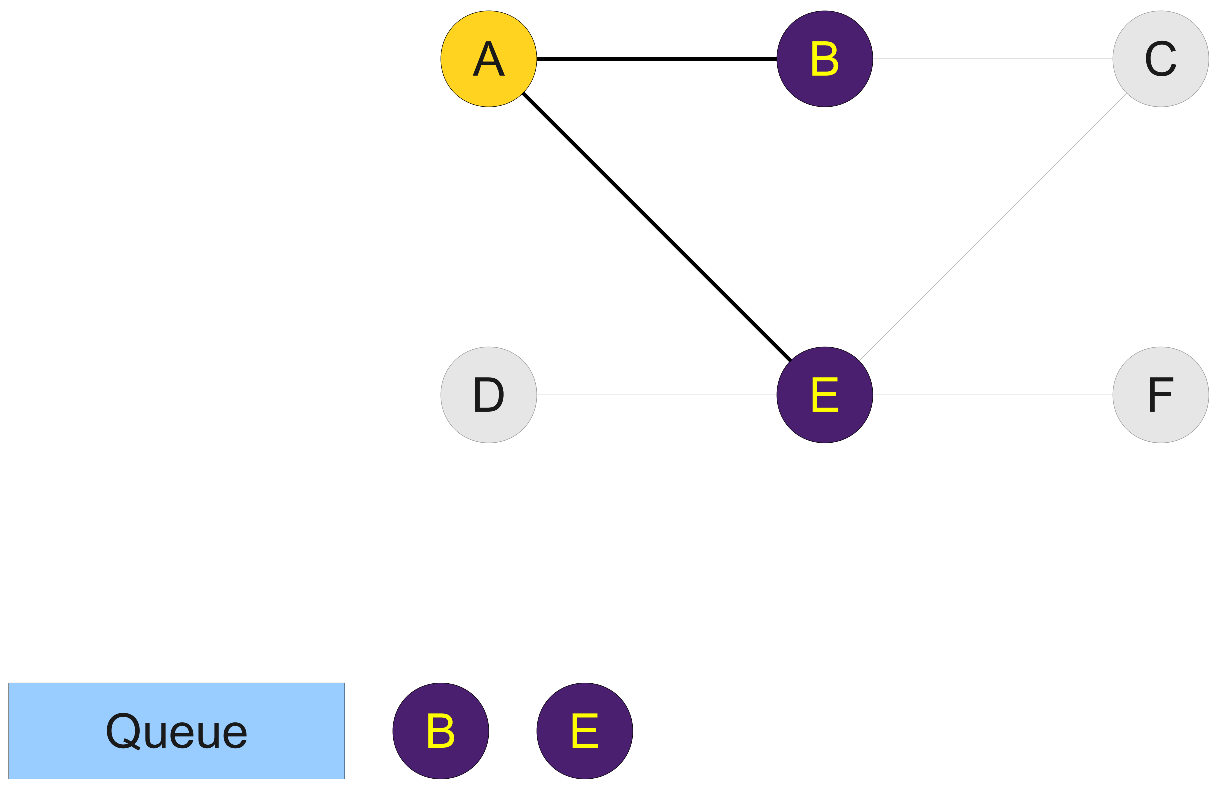


Breadth-first search

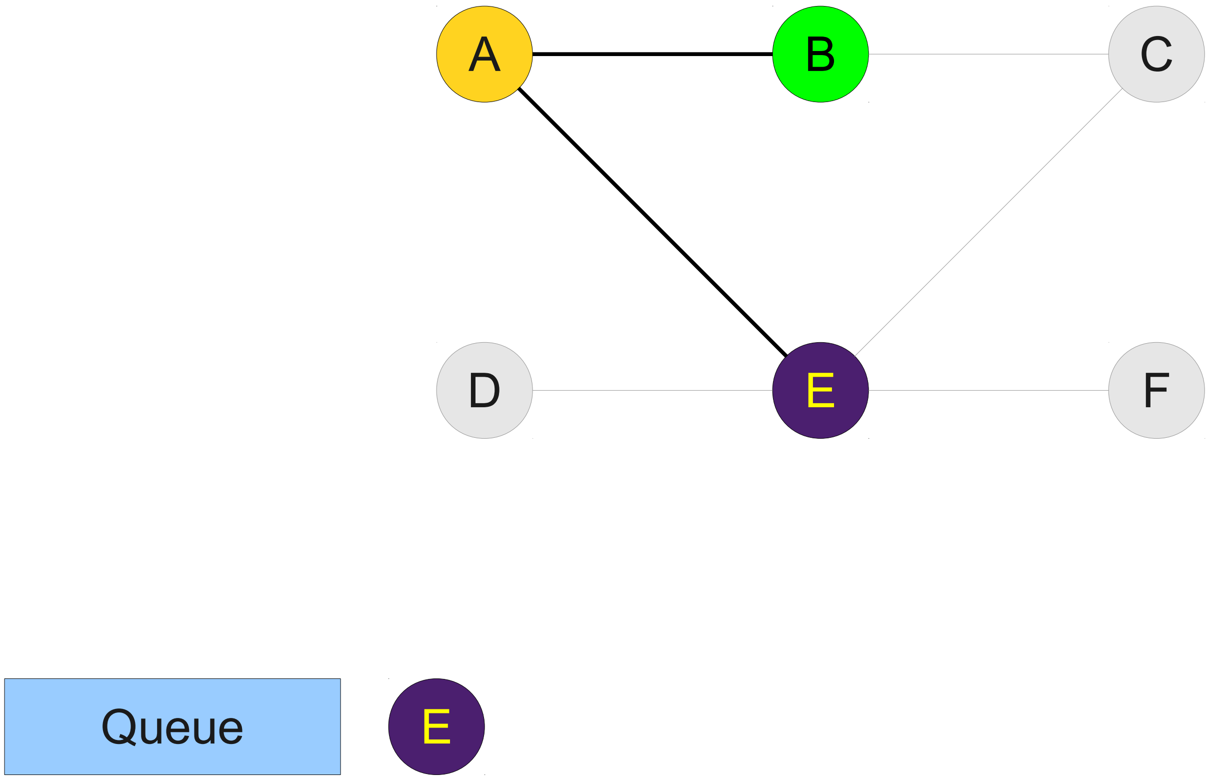




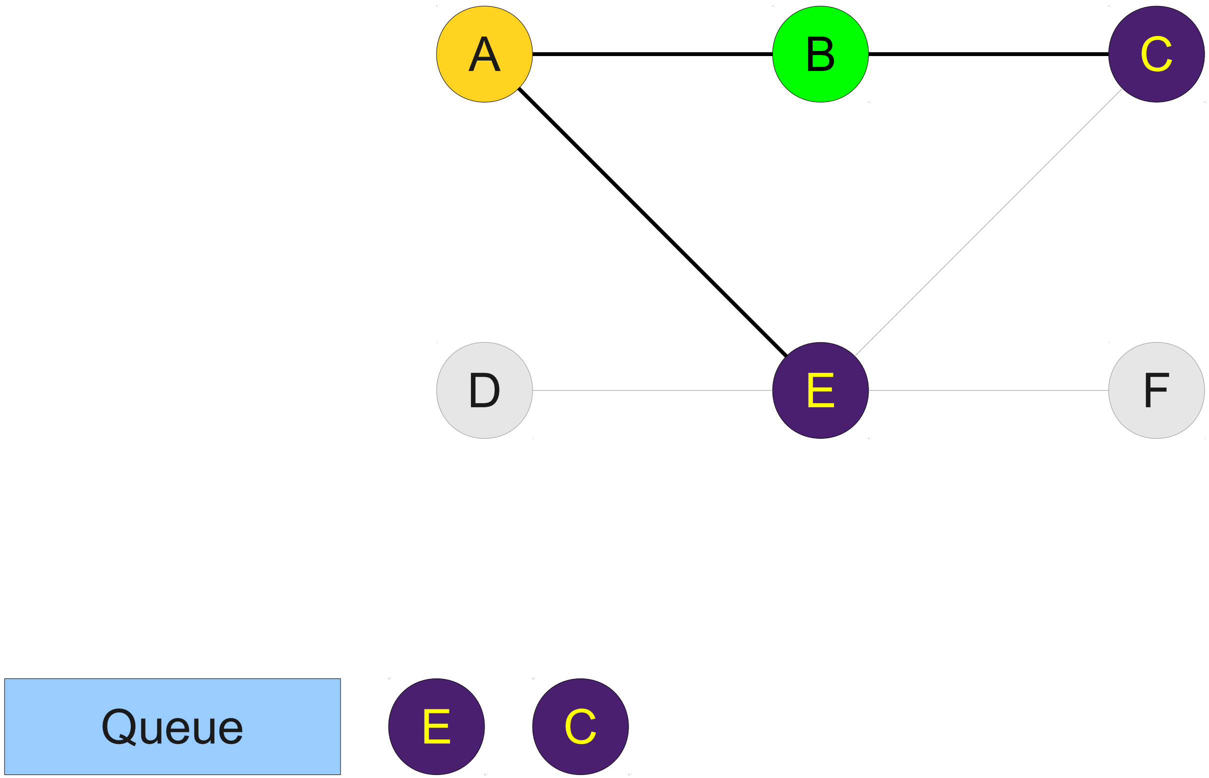
Breadth-first search



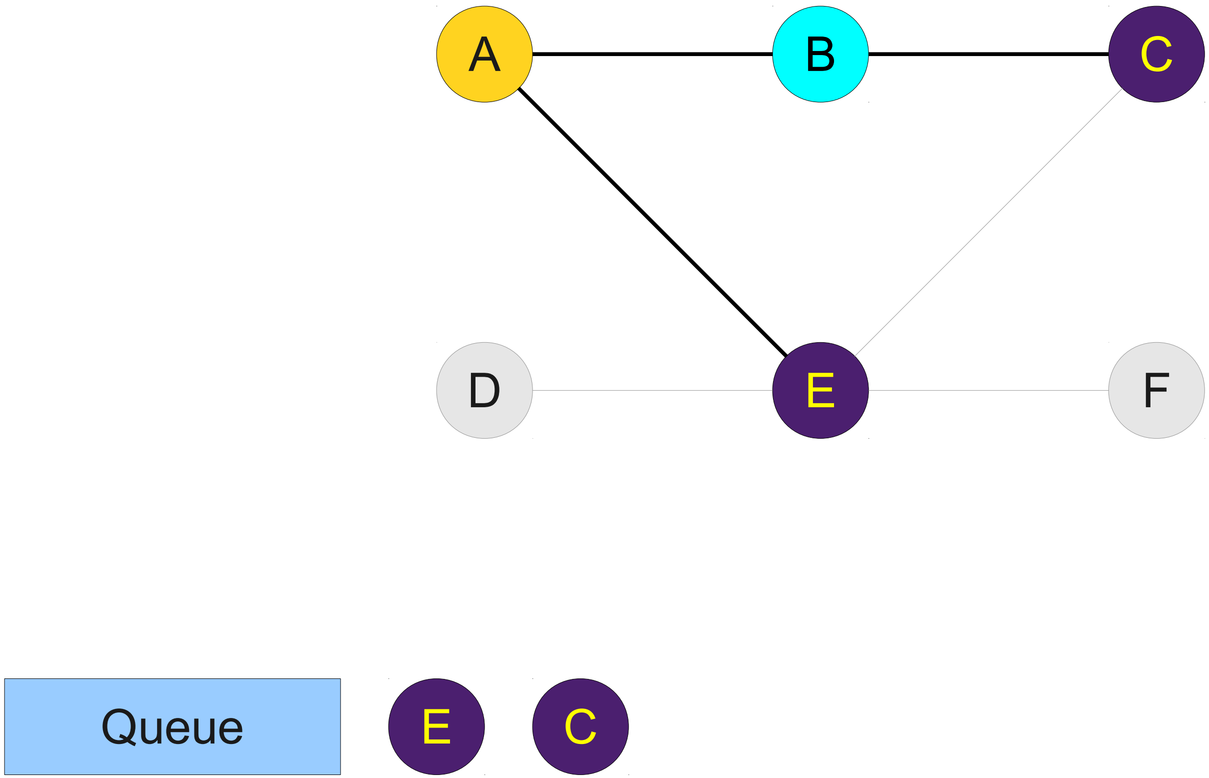
Breadth-first search



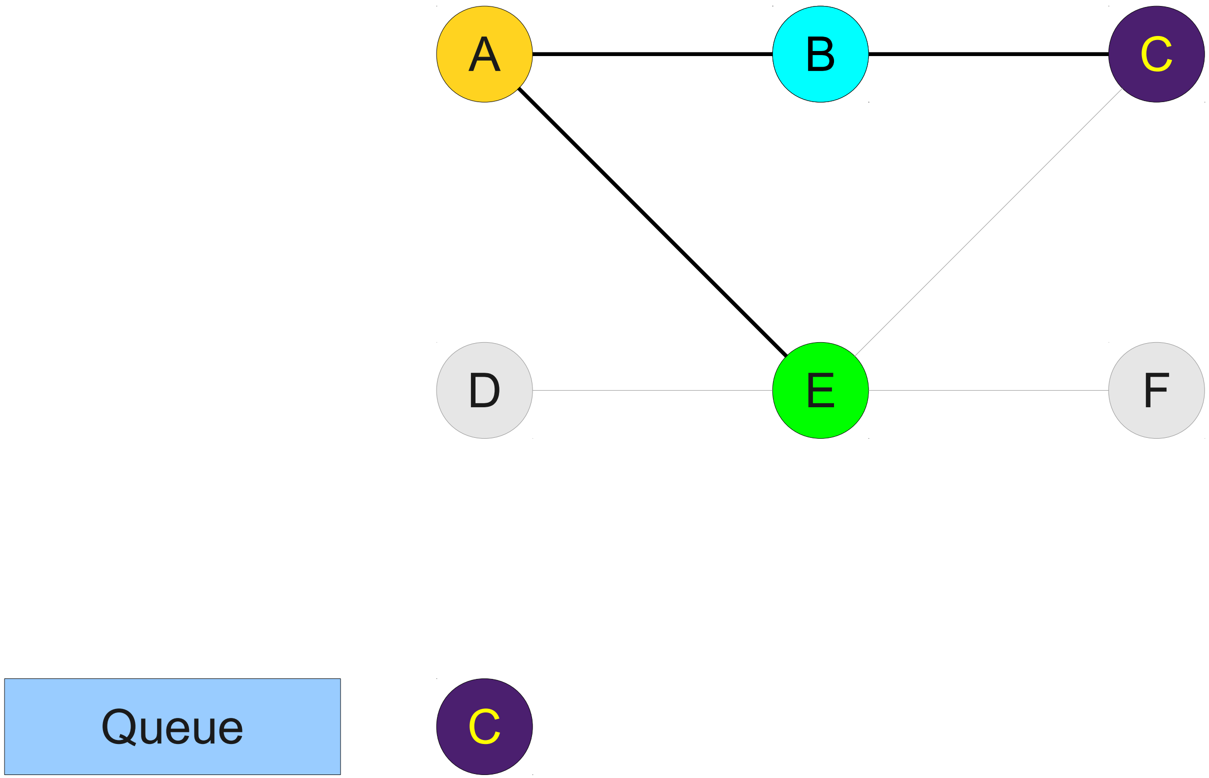
Breadth-first search



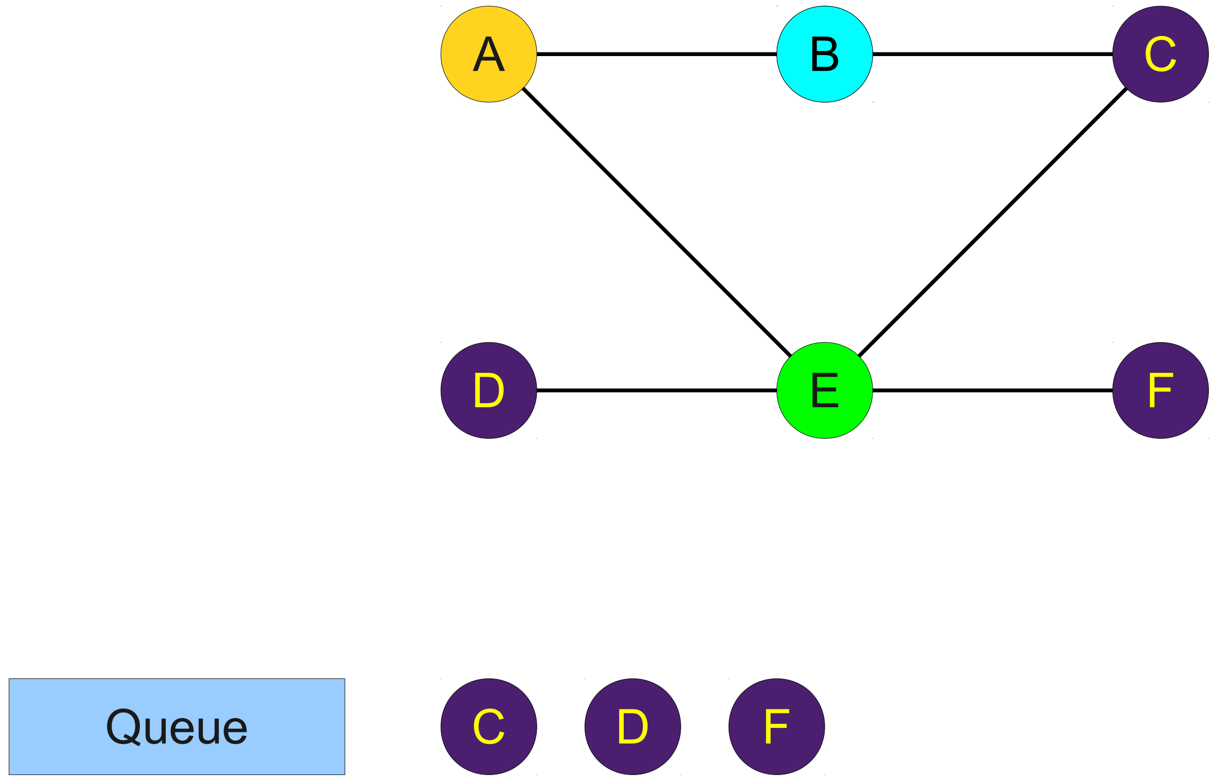
Breadth-first search



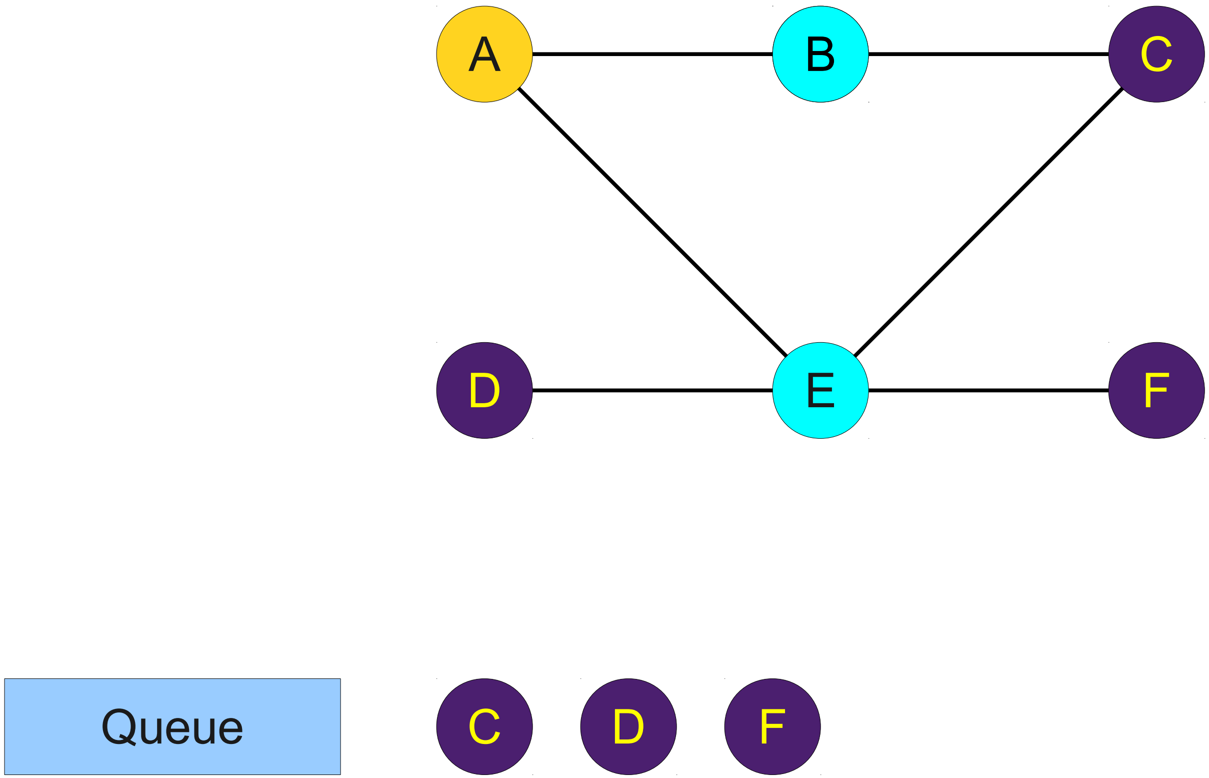
Breadth-first search



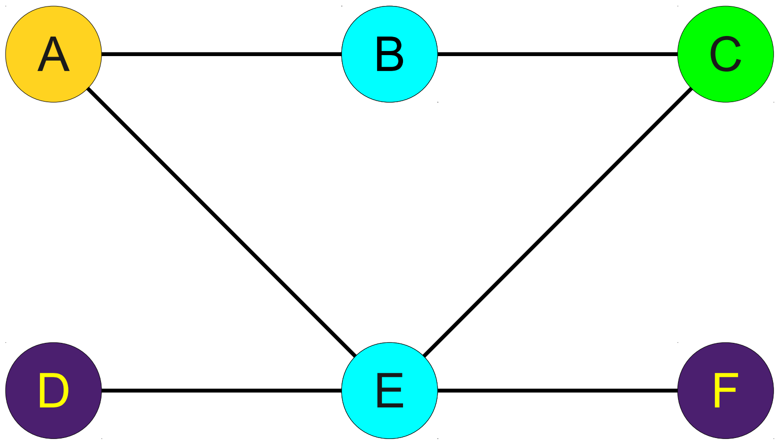
Breadth-first search

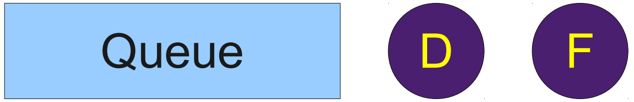


Breadth-first search

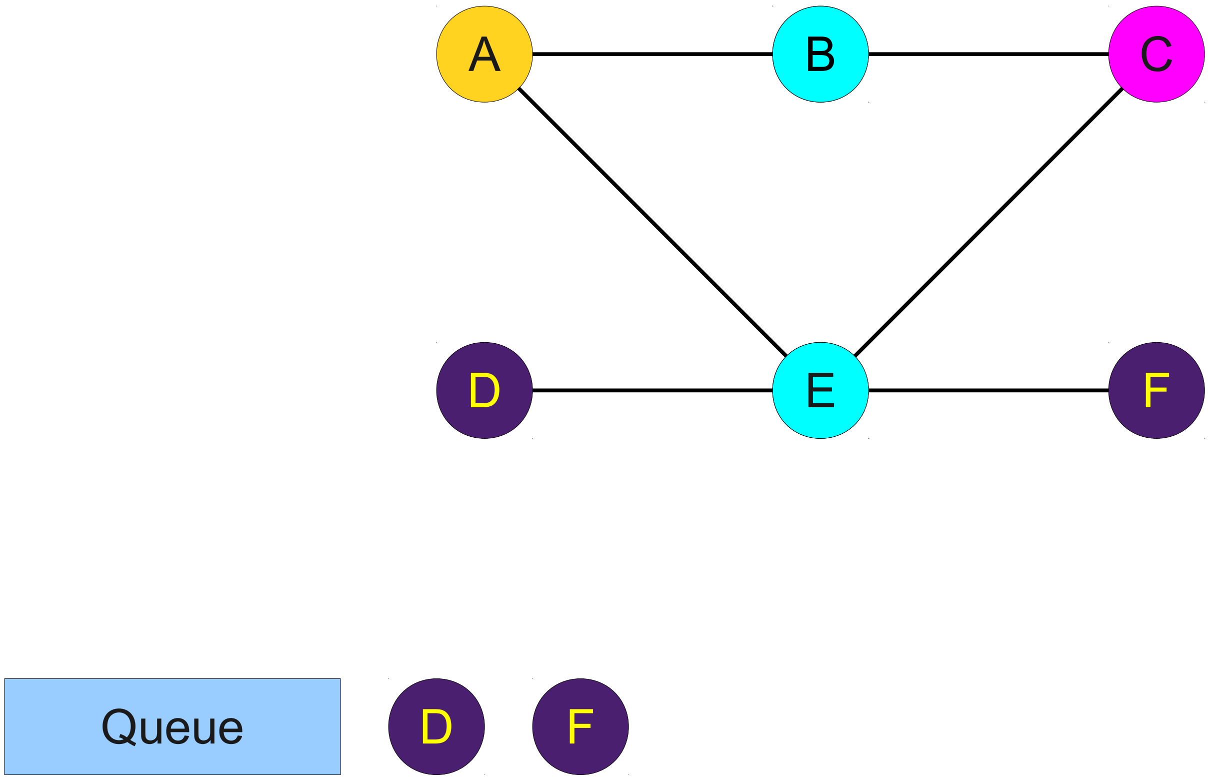


Breadth-first search

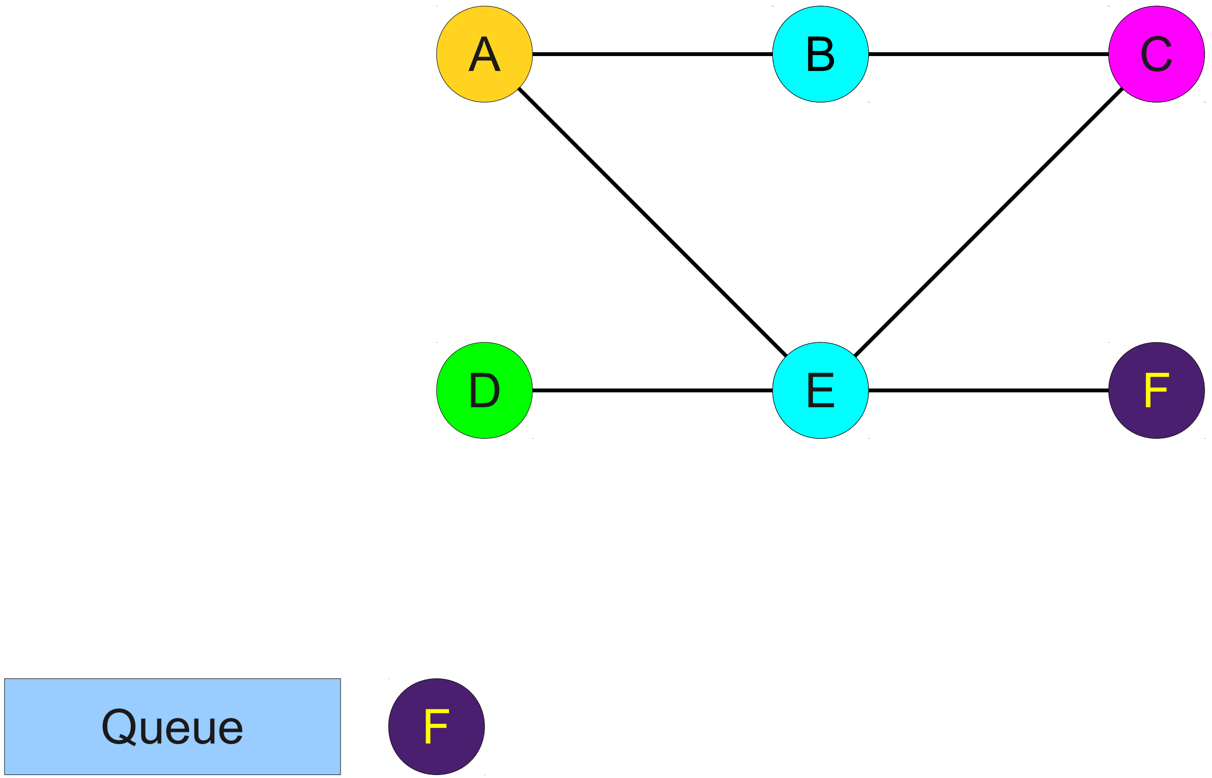




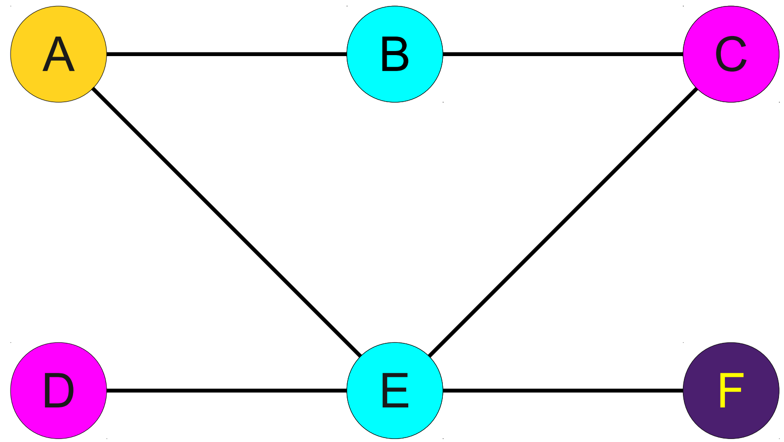
Breadth-first search

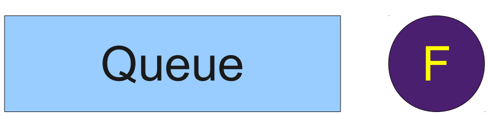


Breadth-first search

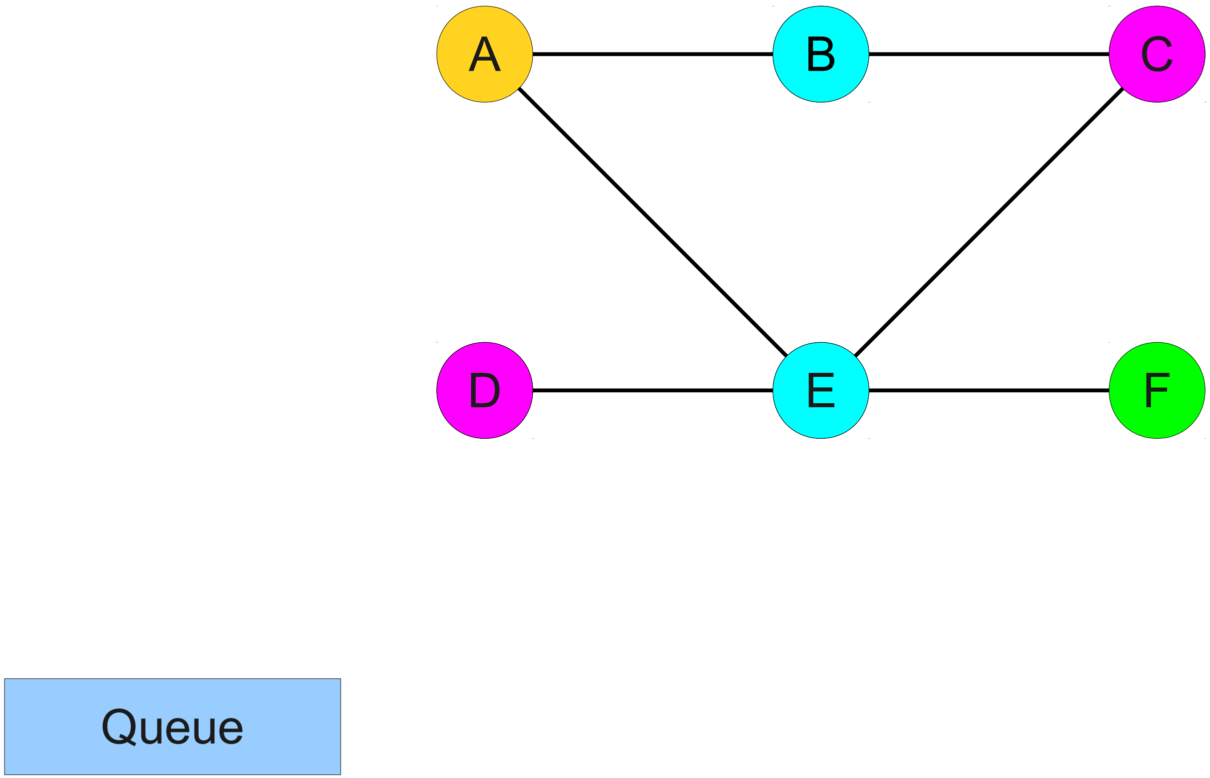


Breadth-first search

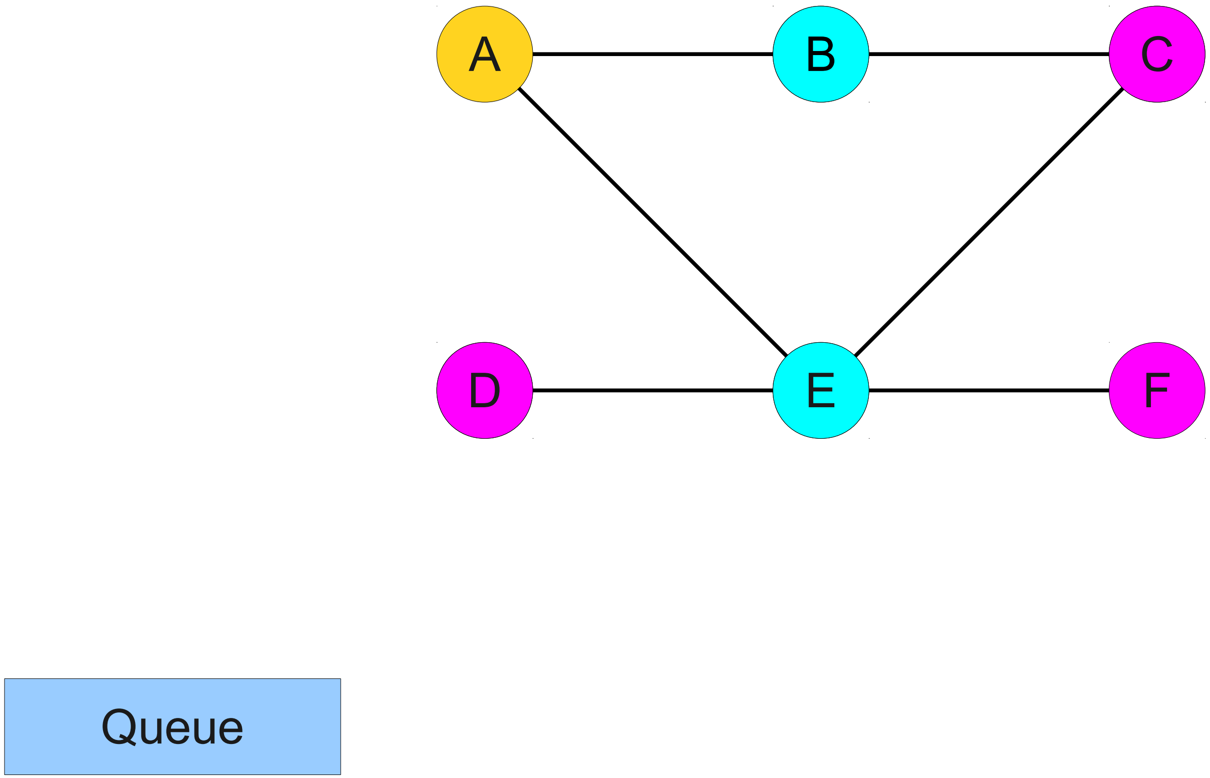




Breadth-first search

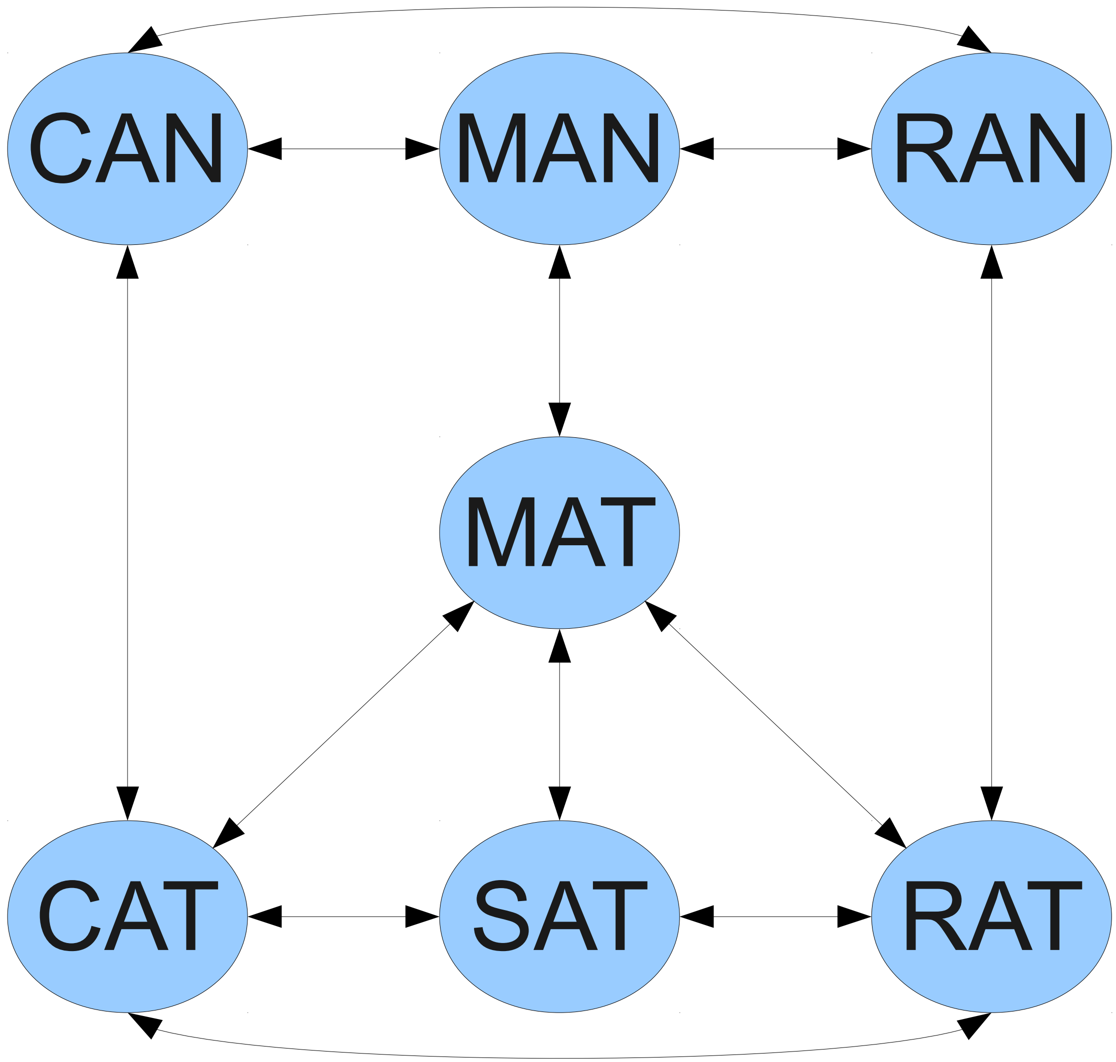


Breadth-first search



Implementing BFS

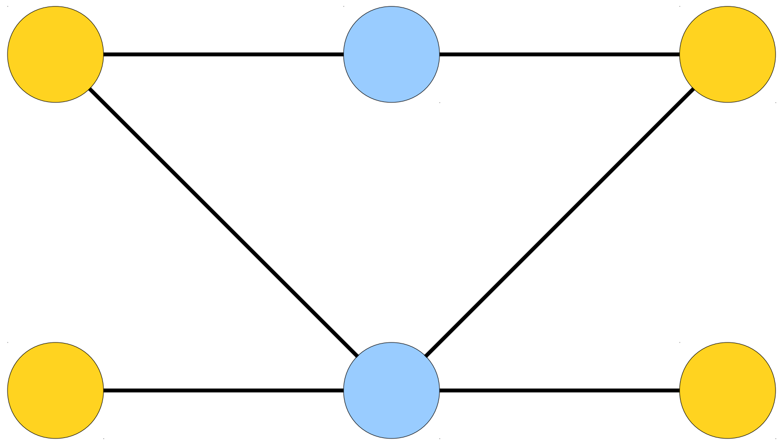
```
BFS(Node v, Set<Node> visited) {
   Create a Queue<Node> of nodes to visit;
   Add v to the queue;

   while (The queue is not empty) {
      Dequeue a node from the queue, let it be u;

      if (u has been visited) continue;
      Add u to the visited set;

      for (Node w connected to u)
         Enqueue w in the queue;
   }
}
```

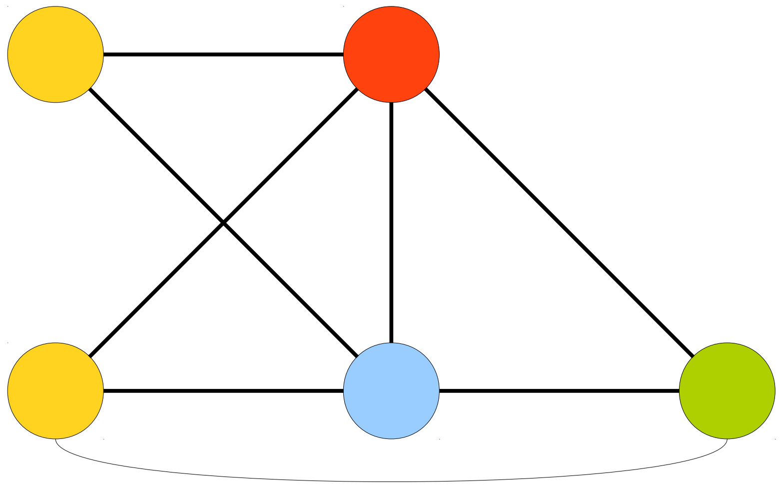

CAN
MAN
RAN
MAT
CAT
SAT
RAT

Classic Graph Algorithms

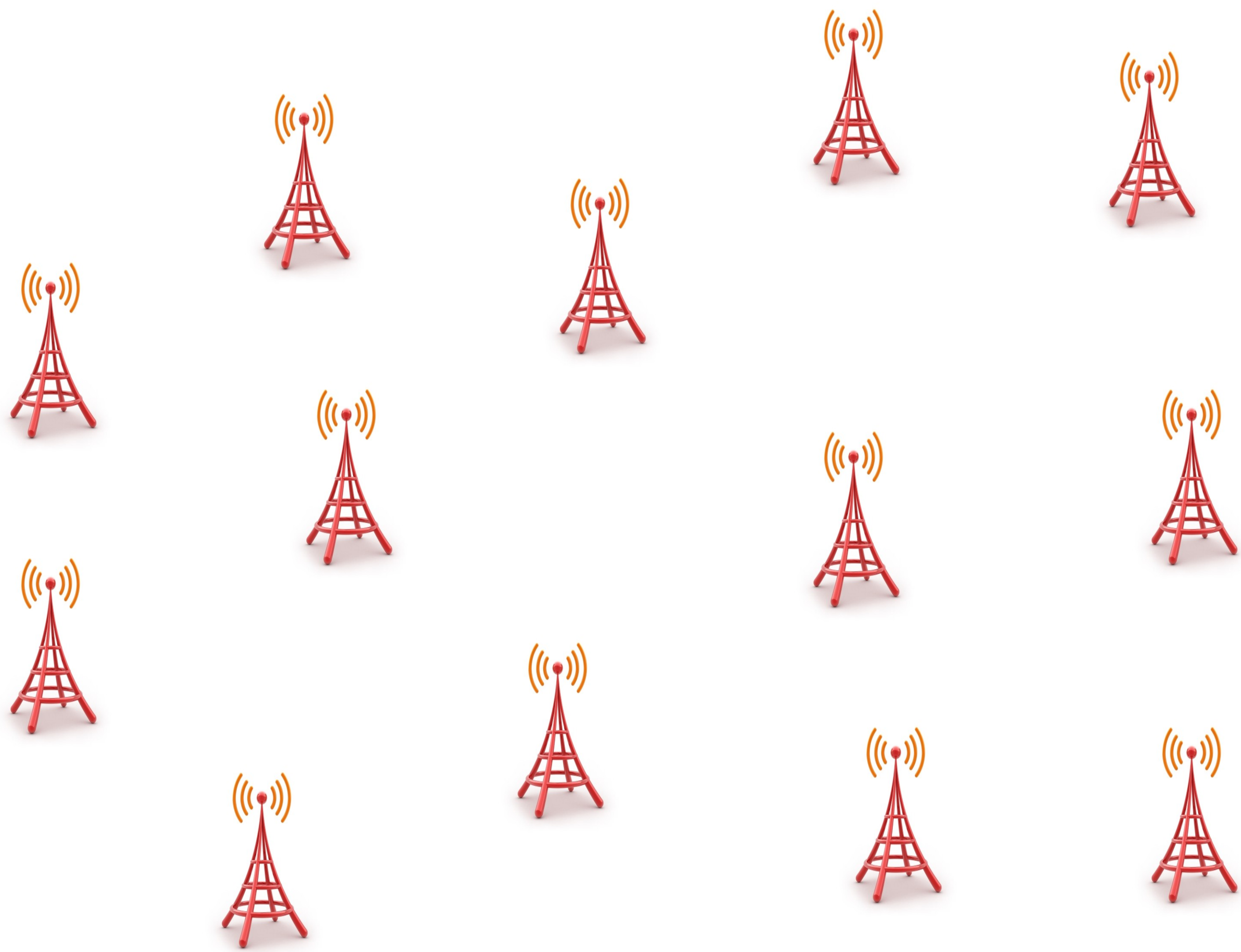
Graph Coloring

- Given a graph G, assign **colors** to the nodes so that no edge has endpoints of the same color.
- The **chromatic number** of a graph is the fewest number of colors needed to color it.

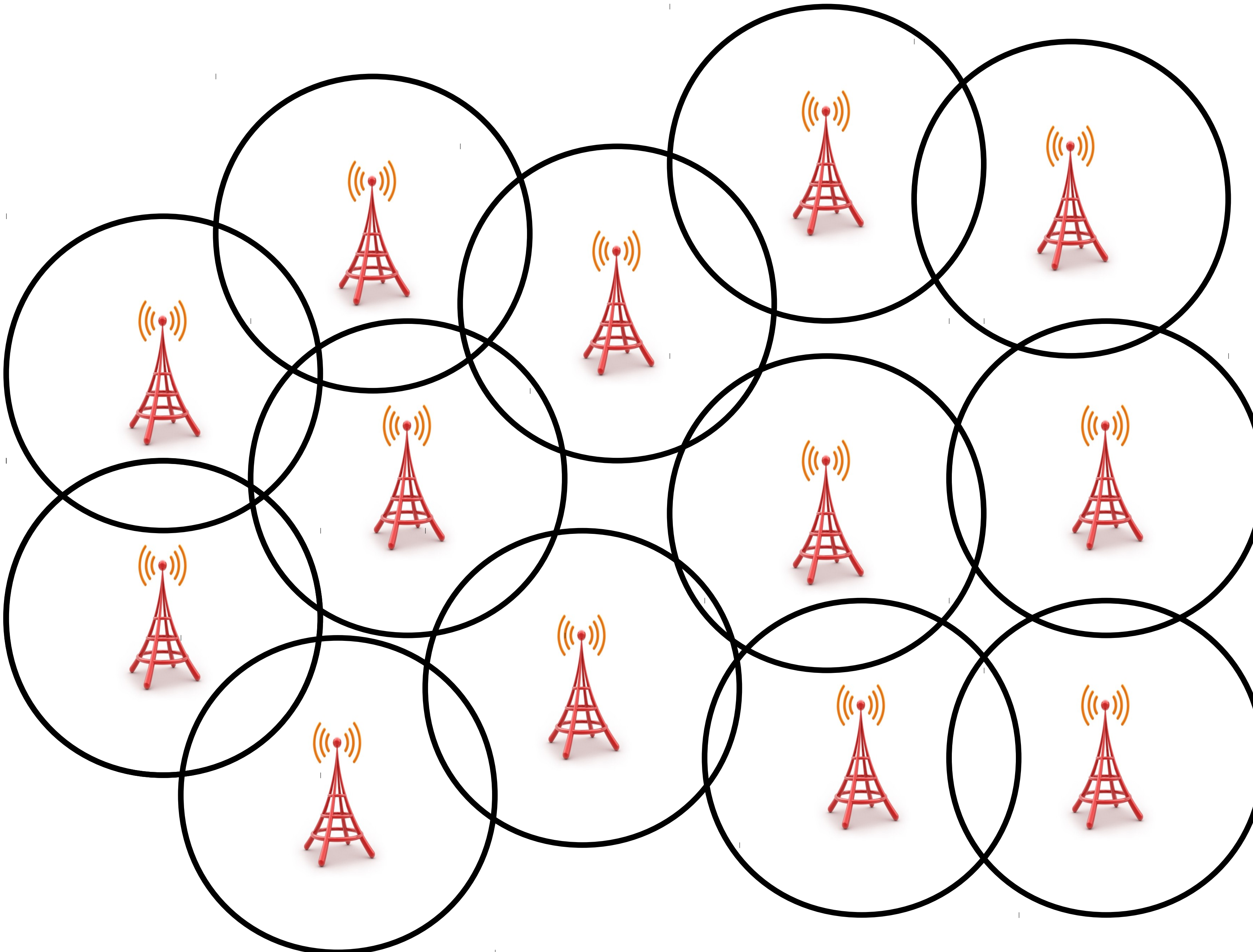




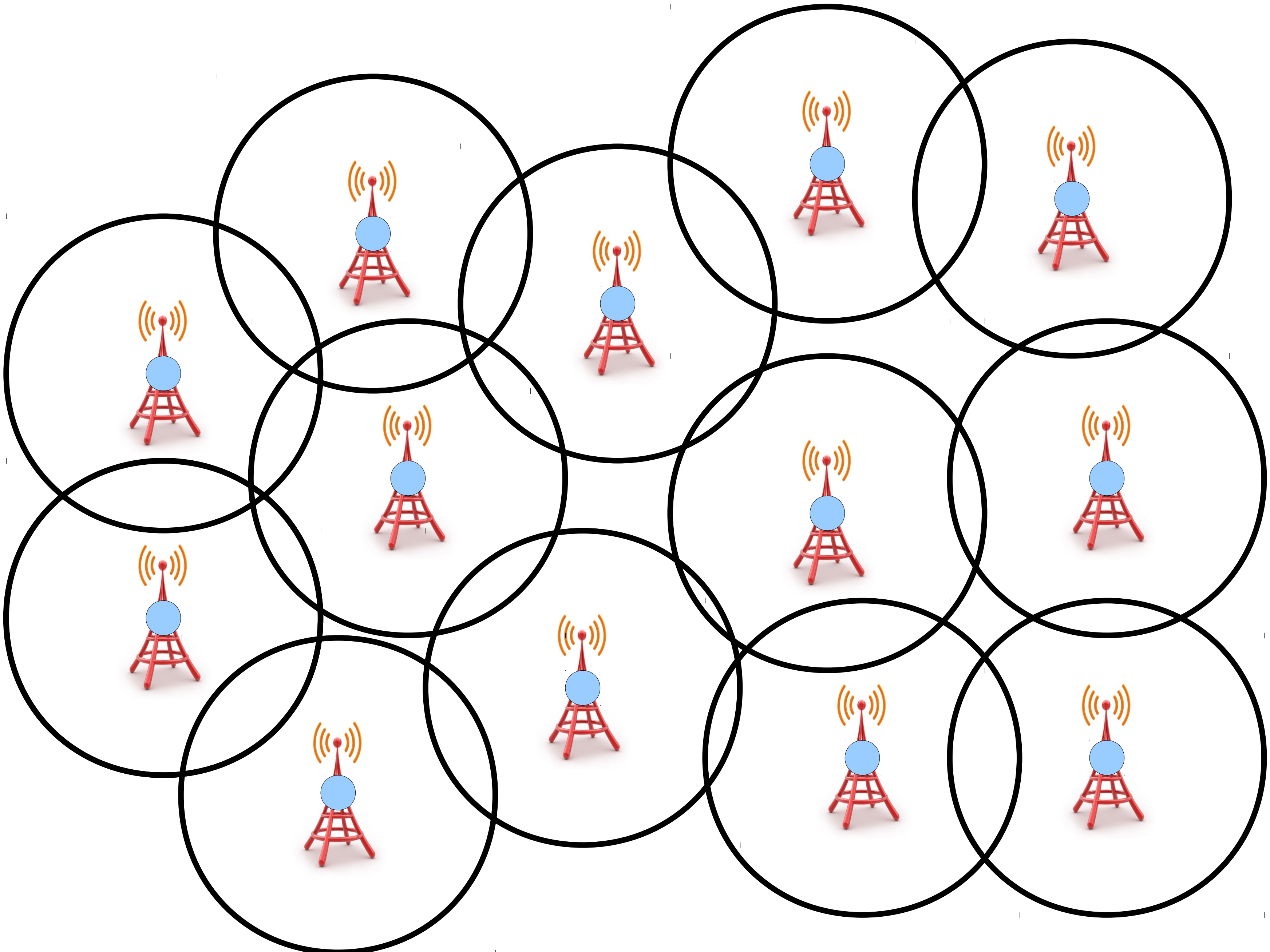
Graph Coloring is Useful



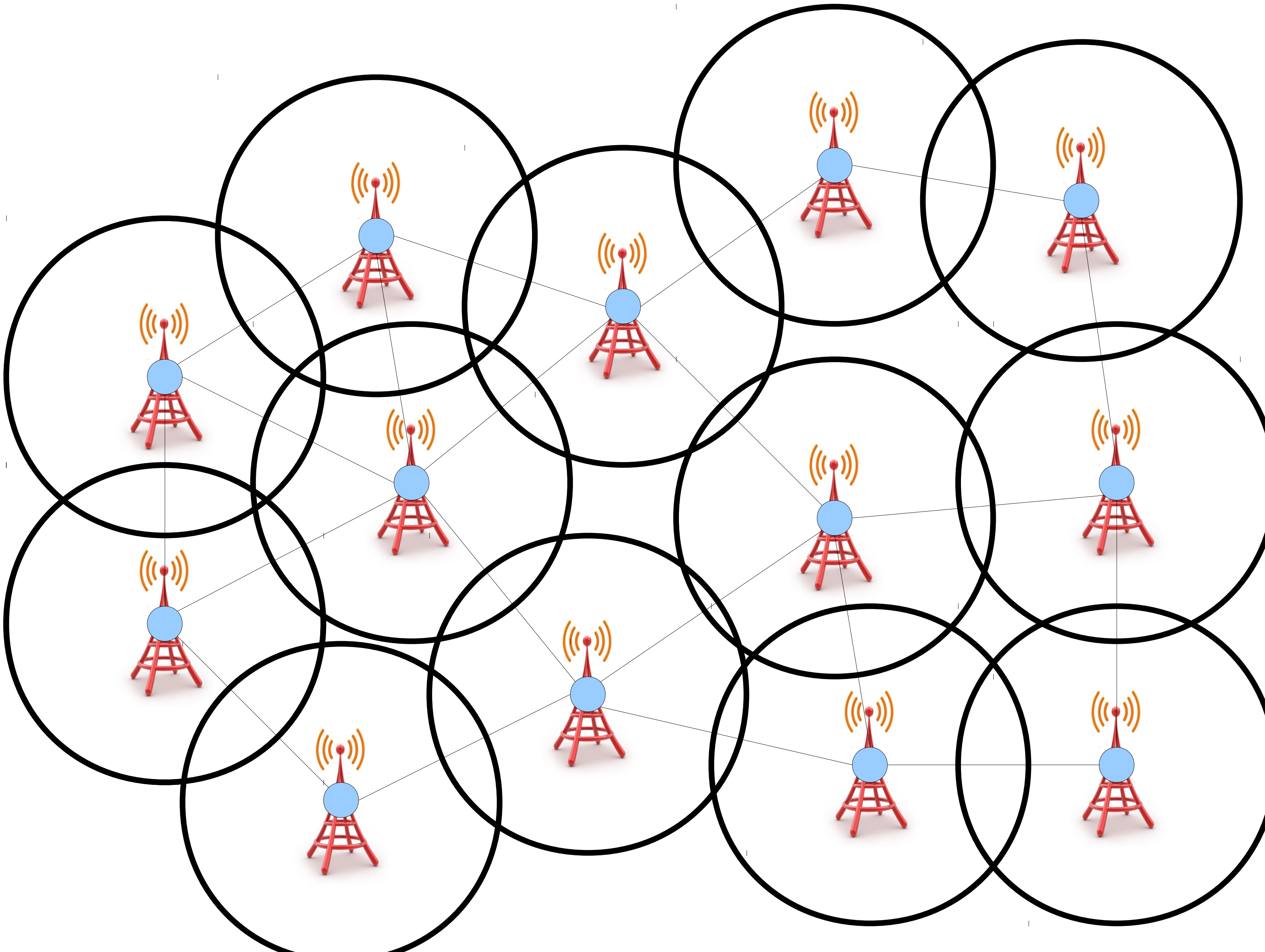
Graph Coloring is Useful



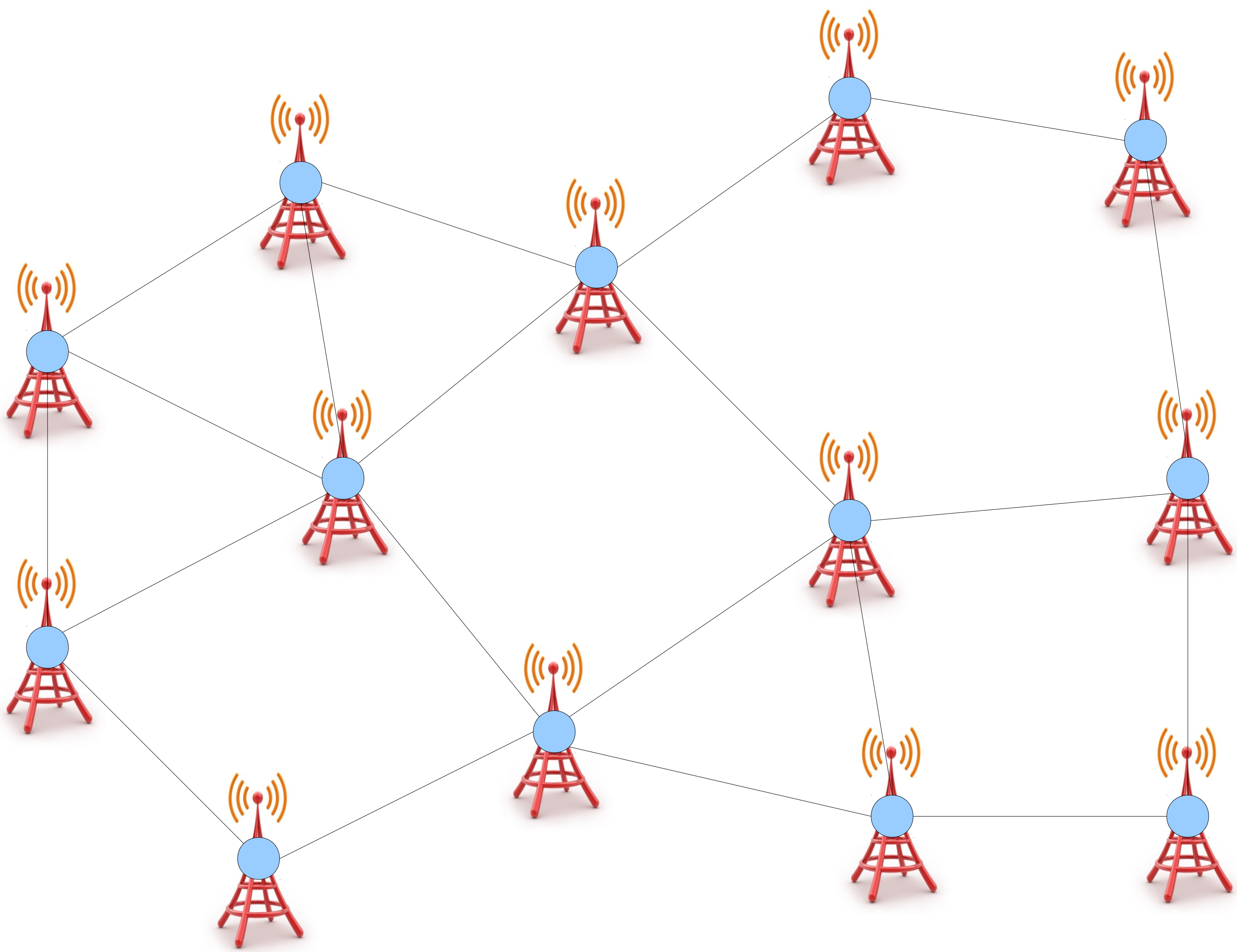
Graph Coloring is Useful



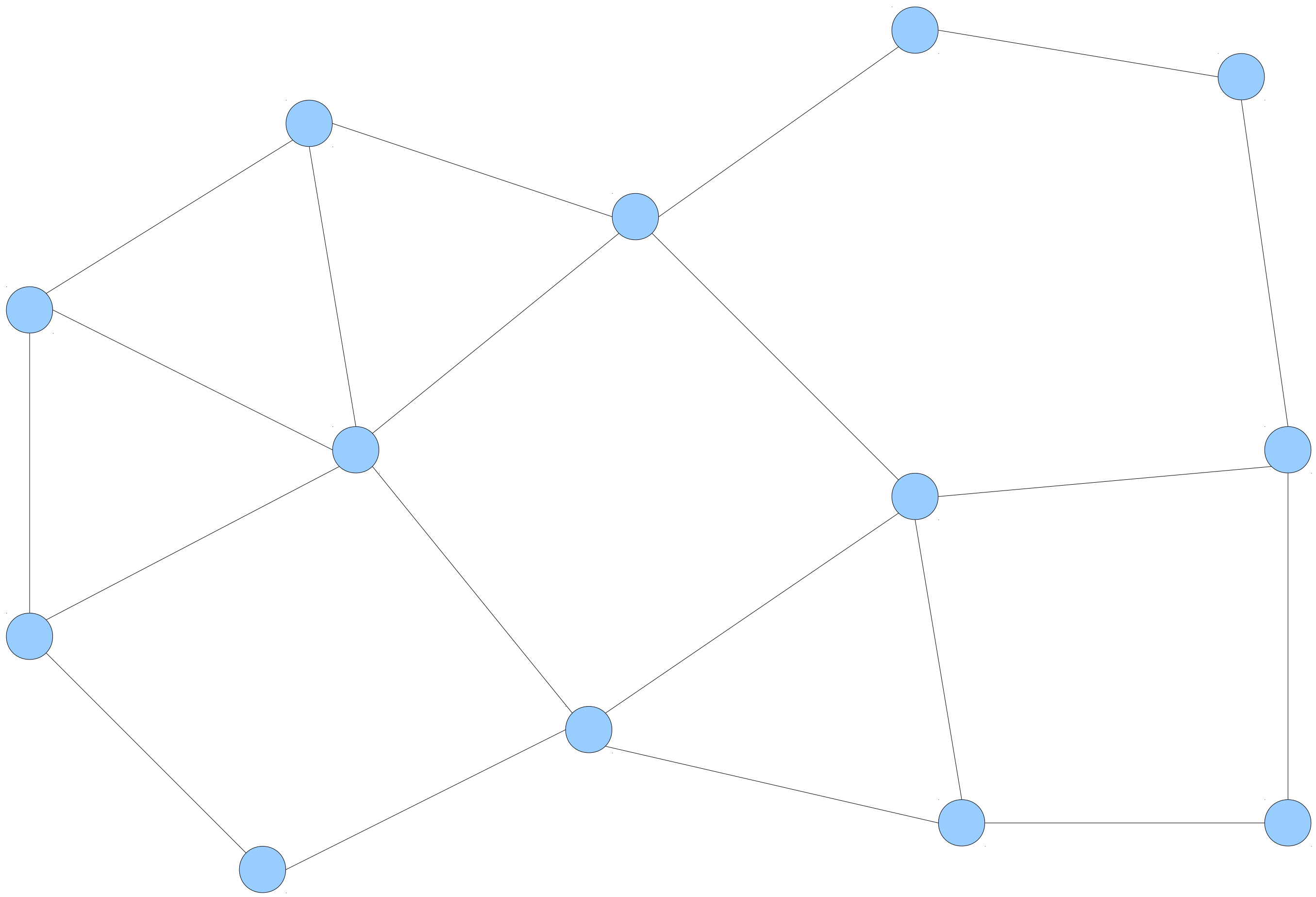
Graph Coloring is Useful



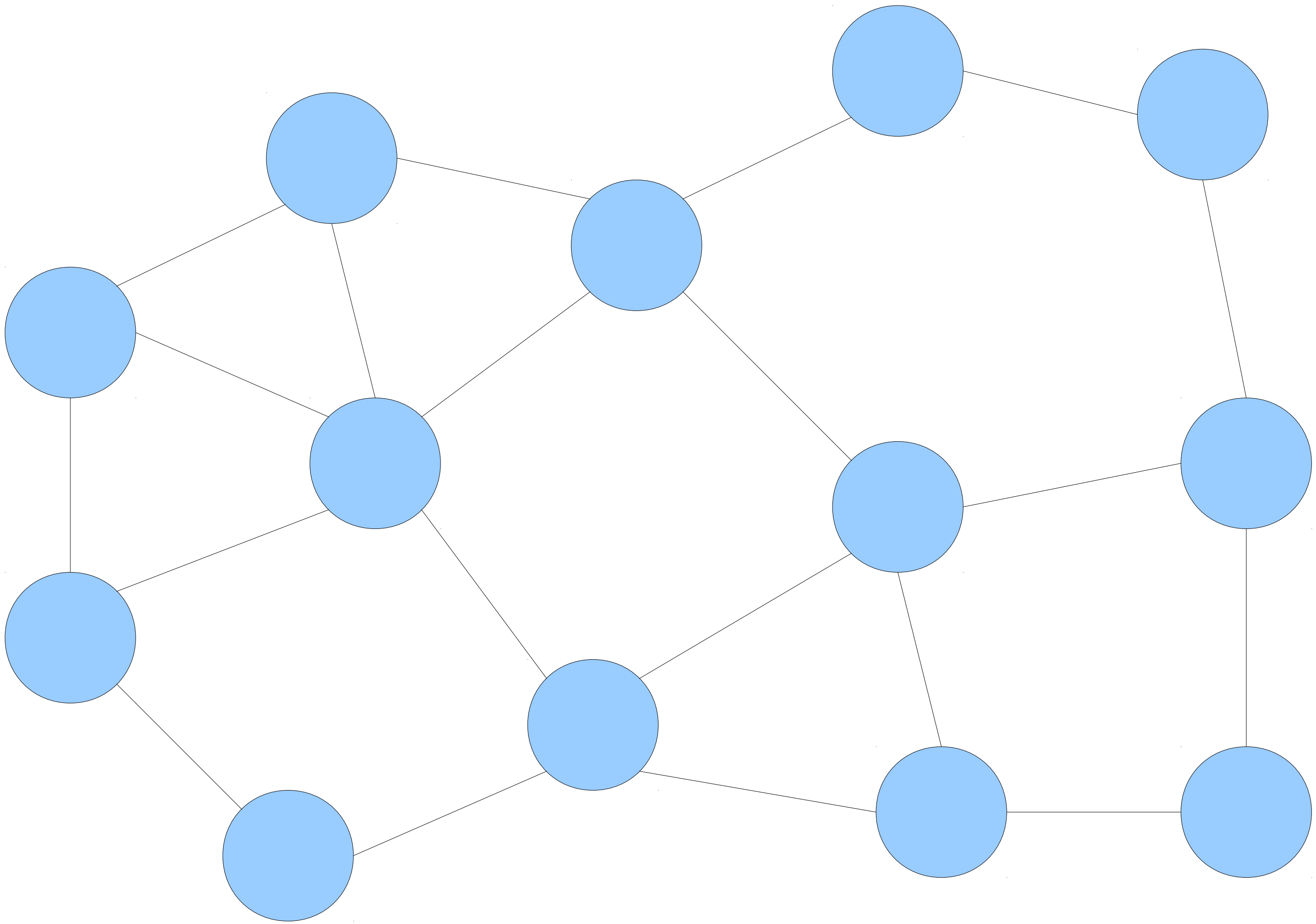
Graph Coloring is Useful



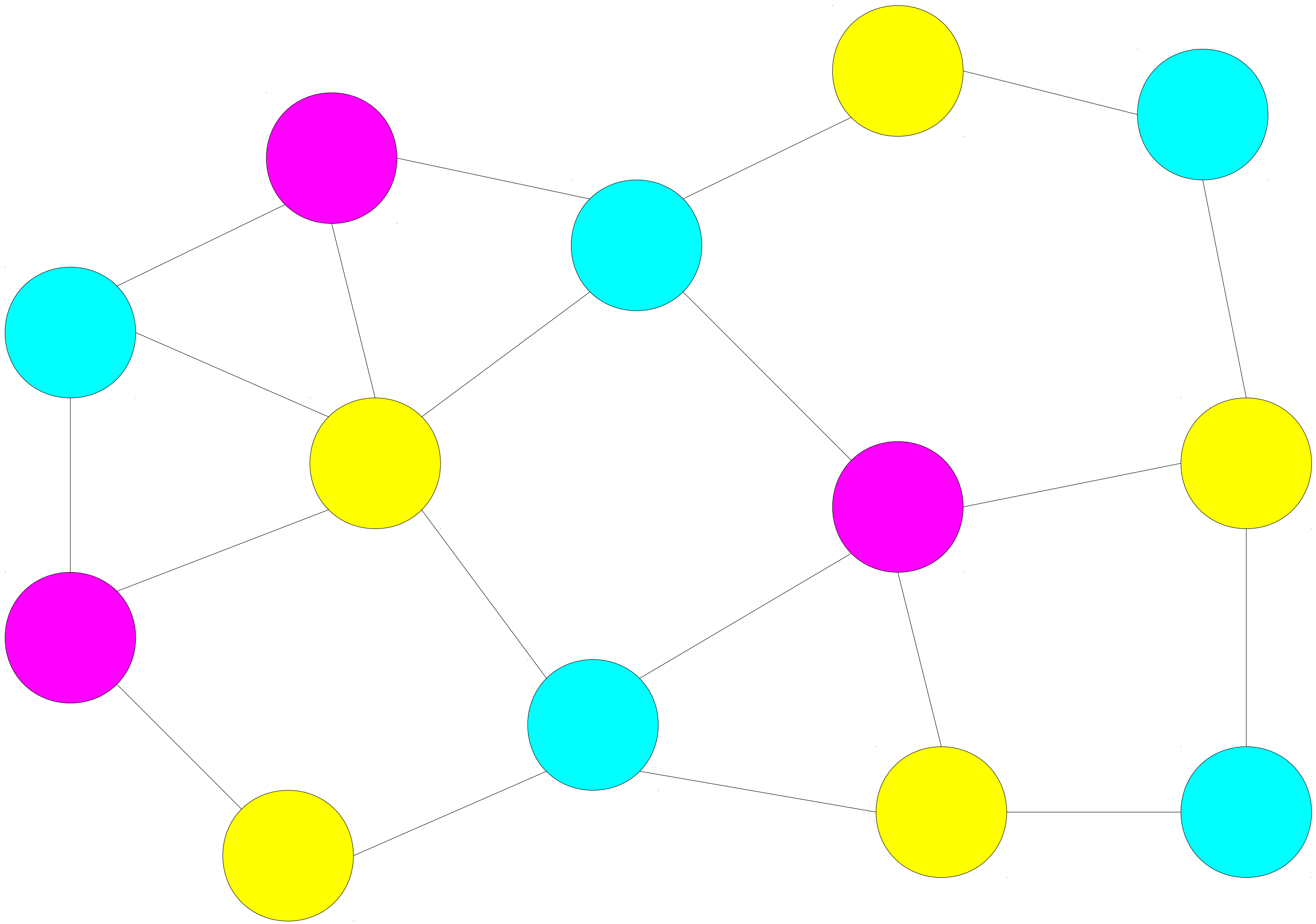
Graph Coloring is Useful



Graph Coloring is Useful



Graph Coloring is Useful



Graph Coloring is **Hard**.

- No efficient algorithms are known for determining whether a graph can be colored with k colors for any $k > 2$.
- Want \$1,000,000? **Find a polynomial-time algorithm** or **prove that none exists**.

Next Time

- **More Graphs**
  - Representing graphs with extra information.
  - Dijkstra's algorithm.
  - Kruskal's algorithm.