Graphs

```
Announcements()
{
```

Change in Schedule

- I'm changing the schedule around a bit...
  - Today: Graphs
  - Tuesday: Shortest Path Algorithms
  - Wednesday: Minimum Spanning Trees
  - Tursday: Review Session for Midterm II

Final

- Final: Monday, August 12th, 7-10pm
  - Location: Cubberly Auditorium
  - Cumulative (but weighted towards post midterm material)
  - Covers material up through Tuesday
    - SCPD students and students who require special arrangements should email me in the next couple days
  - We do the final “early” so we have time to grade it, get it back to you and resolve any grading issues.

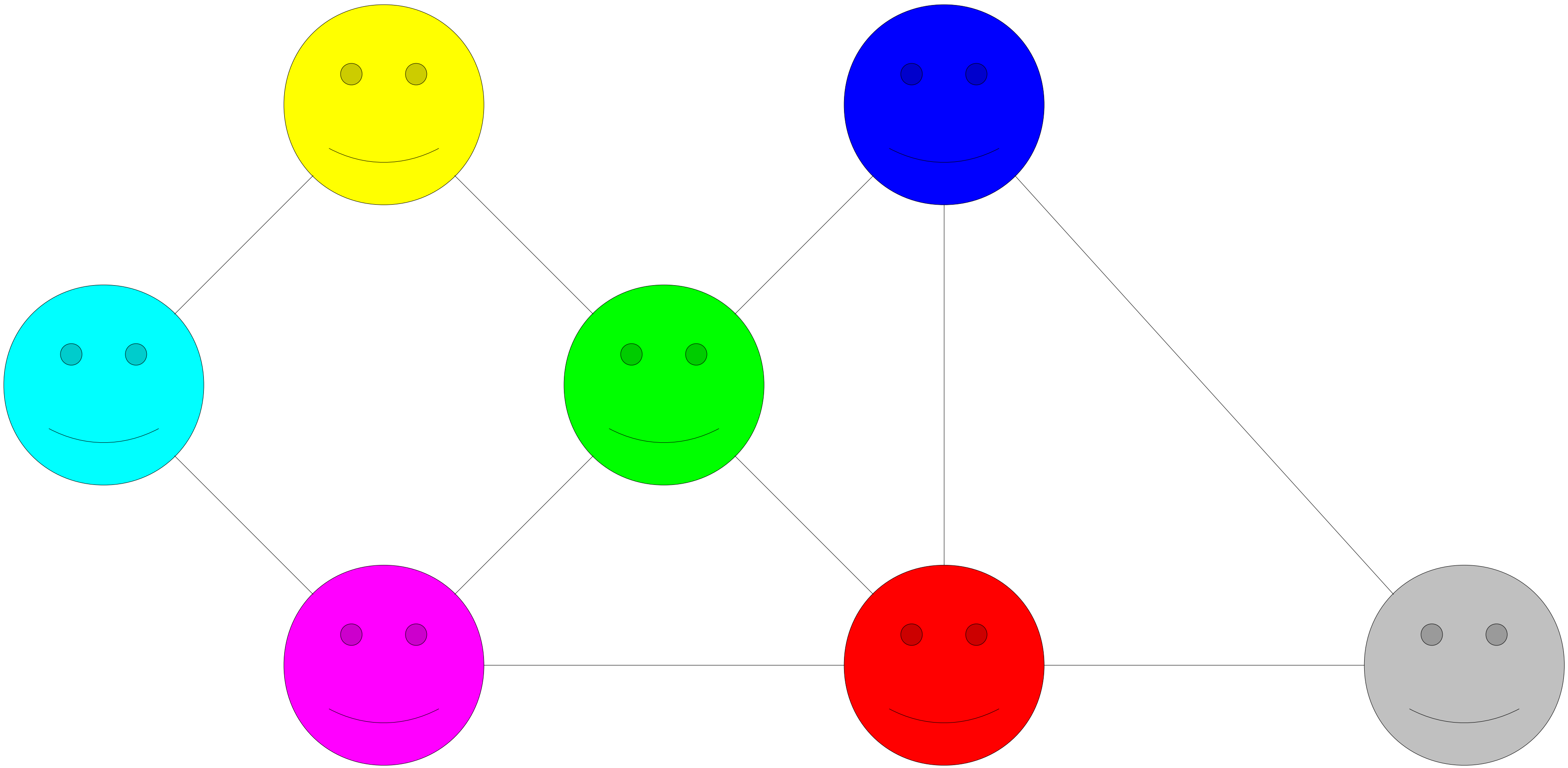
} //Announcements

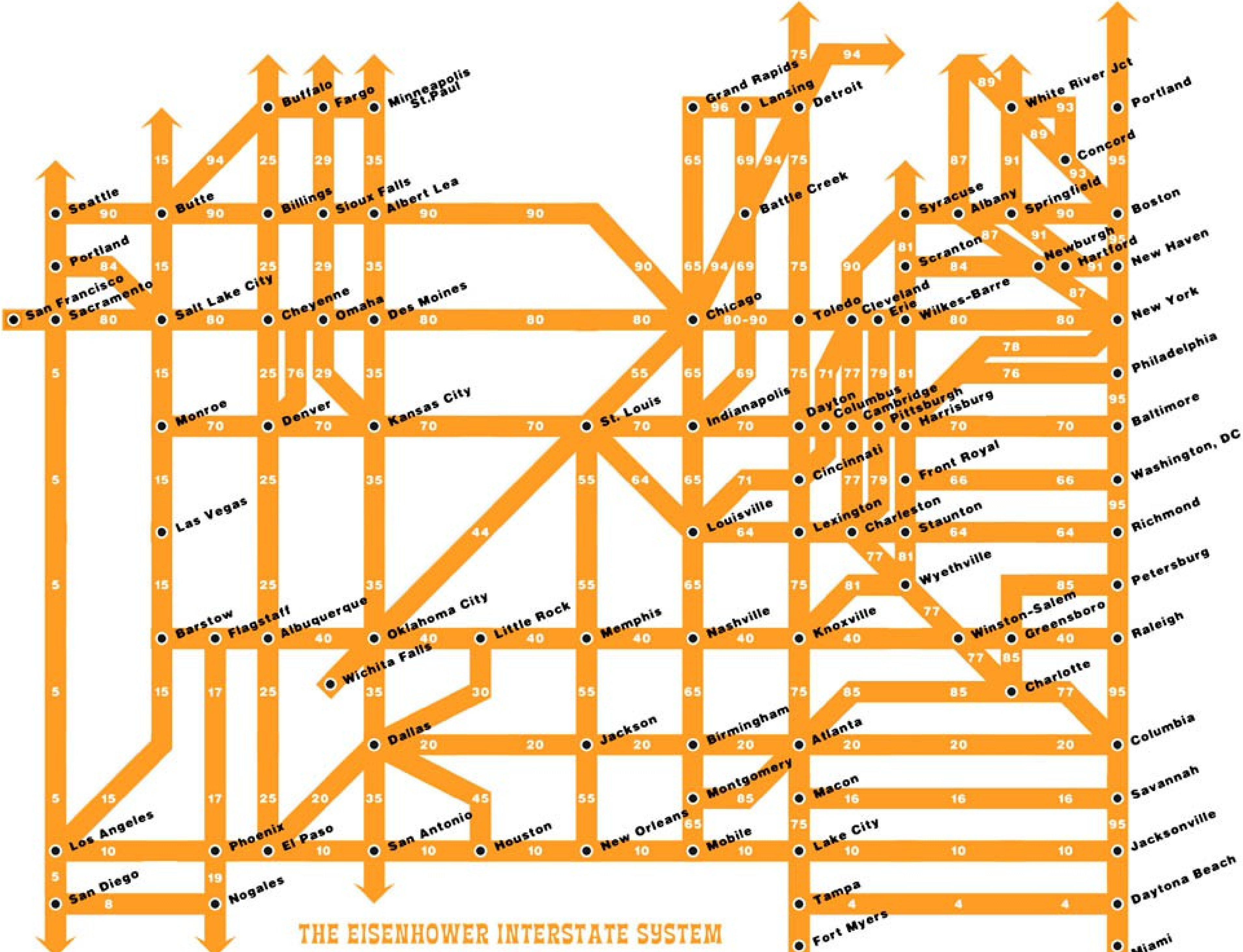
Data Structures Cheat Sheet

- **`Vector`**, **`Stack`**: dynamic array
- **`Queue`**: linked list or dynamic array
- **`Set`**, **`Map`**: Binary Search Tree
- **`HashSet`**, **`HashMap`**: Hash Table
- **`Lexicon`**: Trie

Graphs

A Social Network

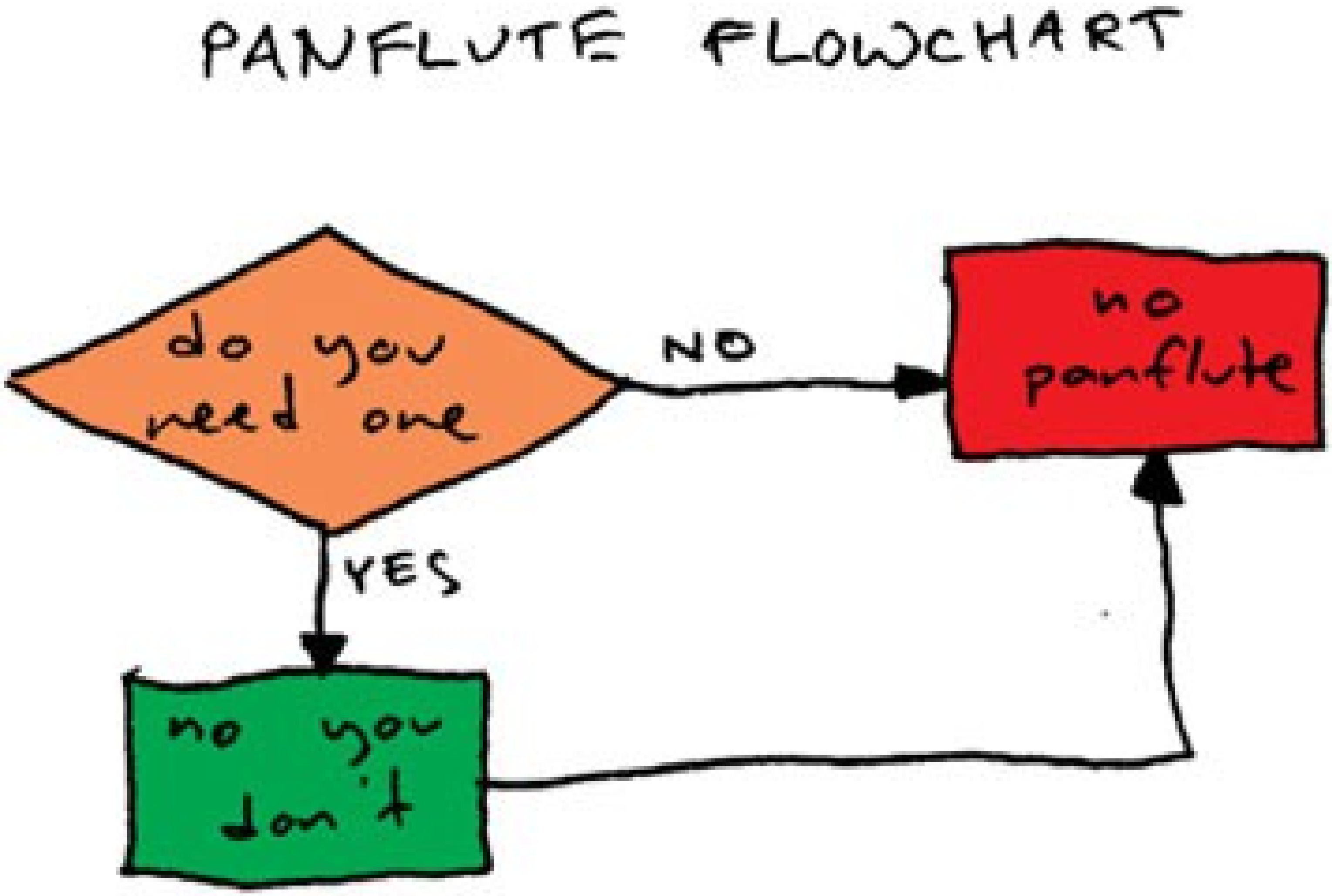




THE EISENHOWER INTERSTATE SYSTEM

(simplified)

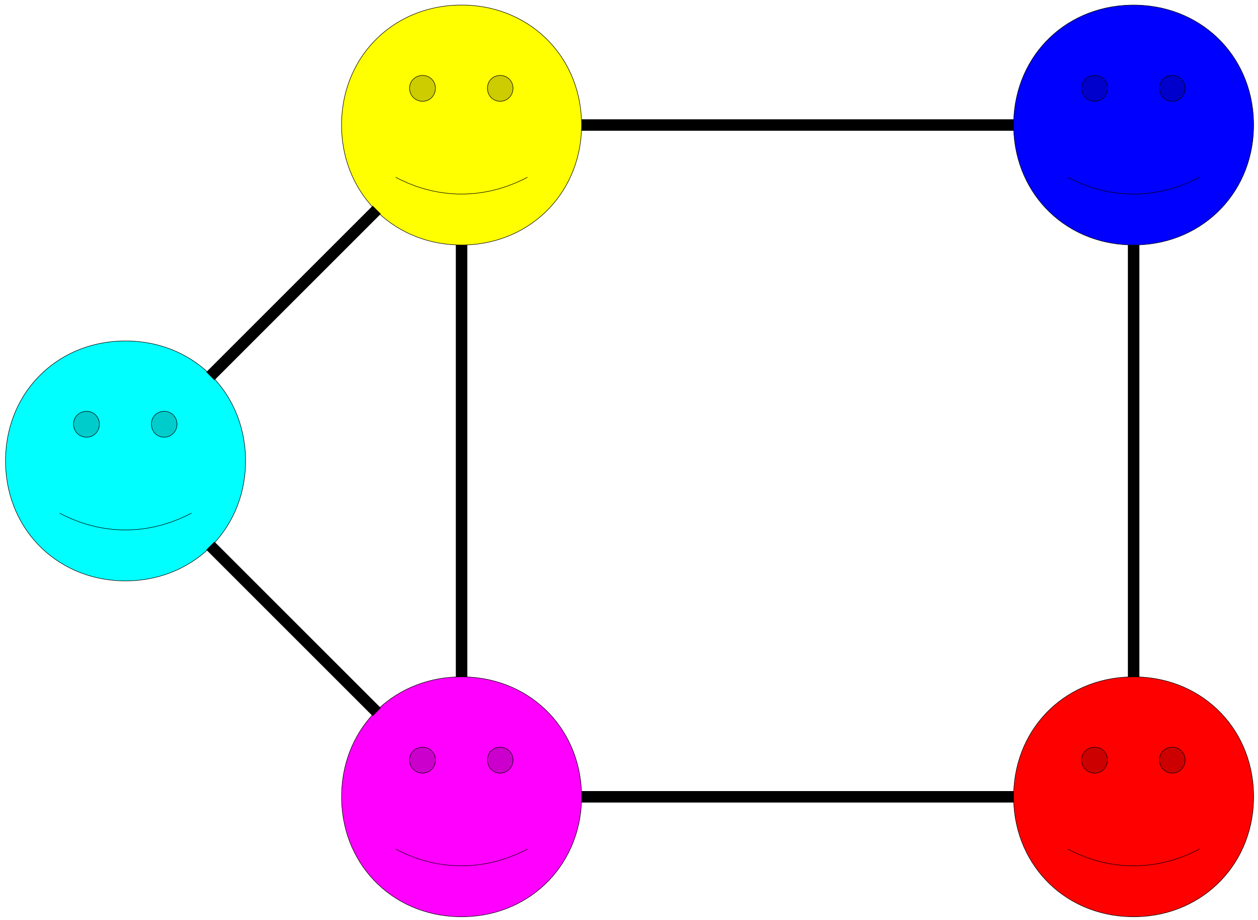
CHRIS YATES 2007


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

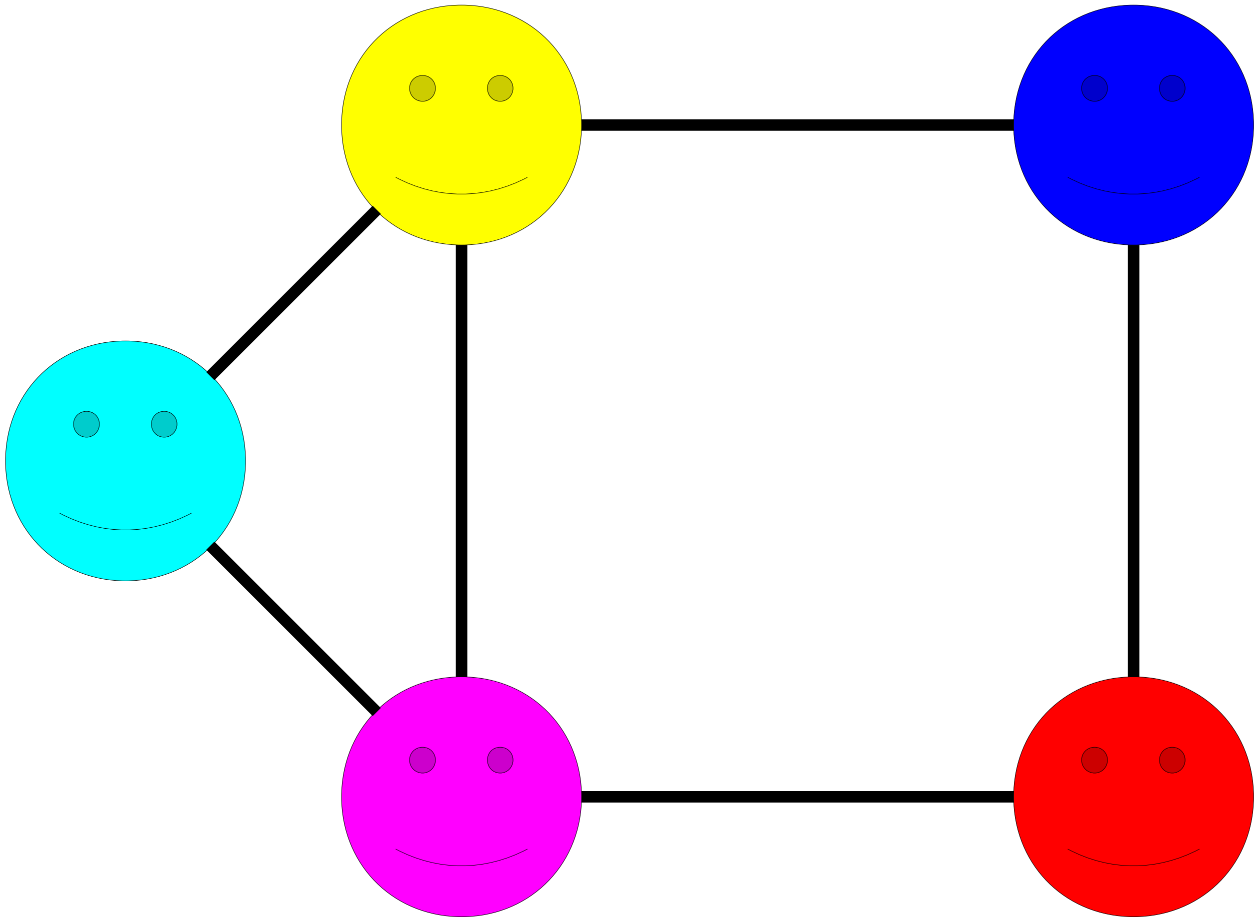
A **graph** is a mathematical structure for representing relationships.

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

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

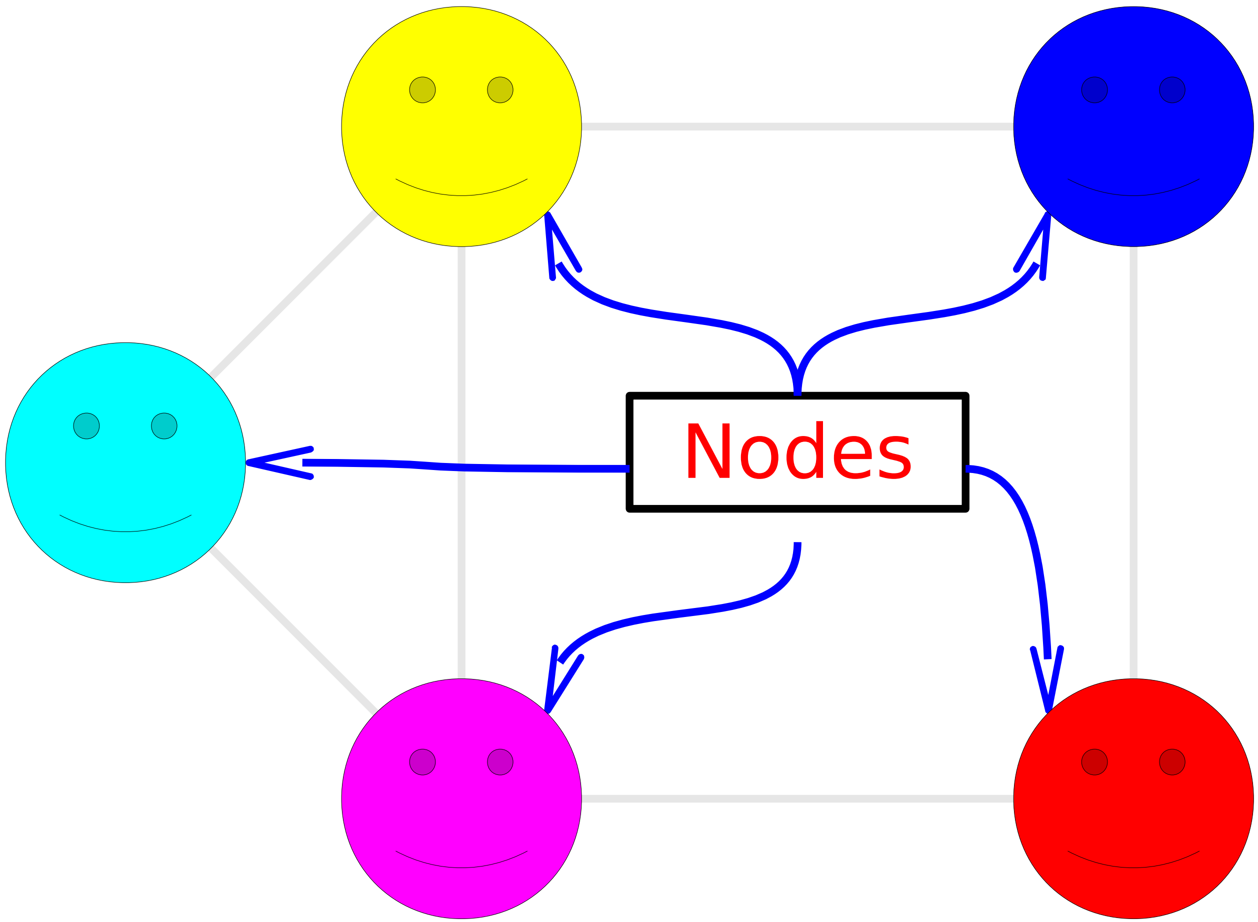


A **graph** is a mathematical structure for representing relationships.



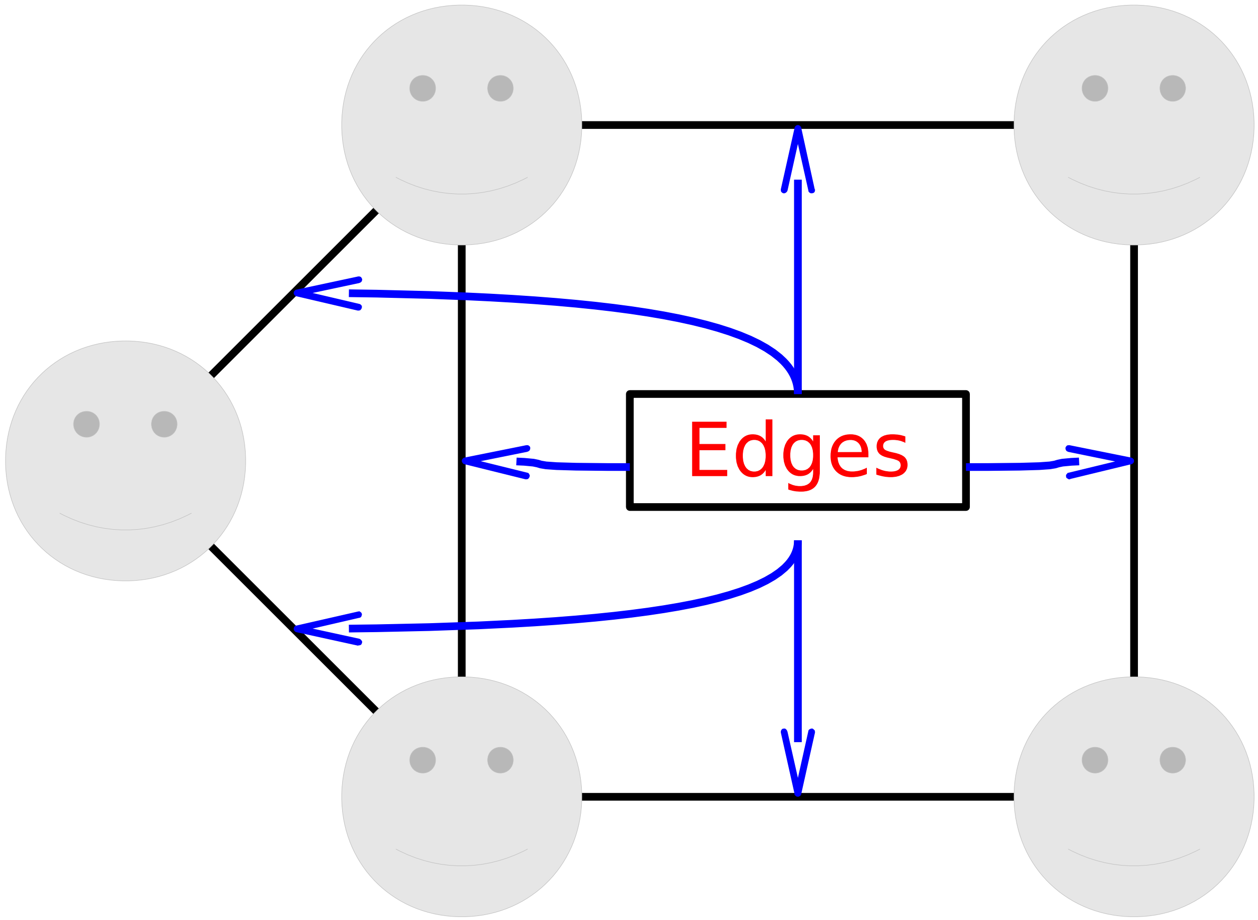
A graph consists of a set of **nodes** connected by **edges**.

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



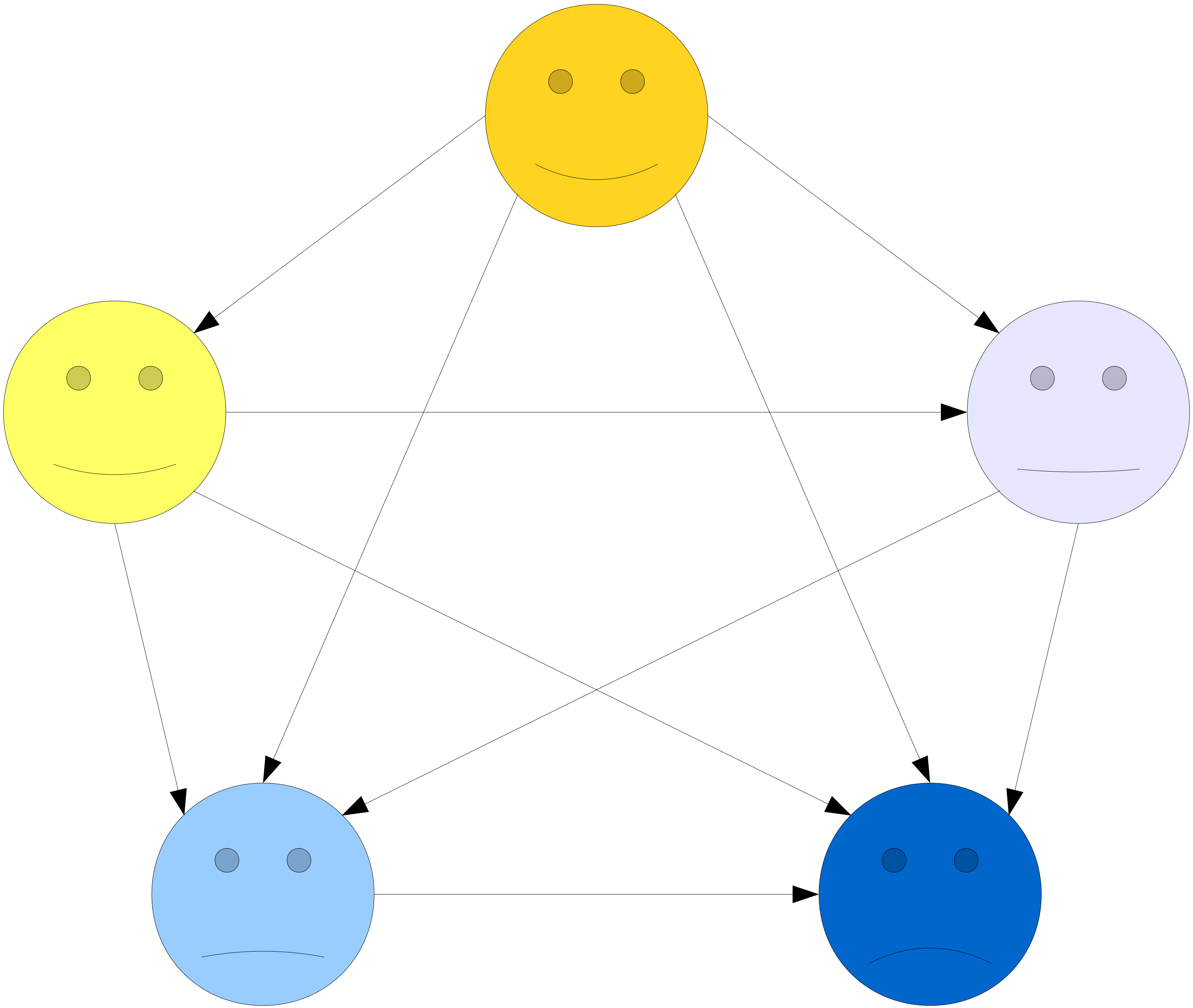
A graph consists of a set of **nodes** connected by **edges**.

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

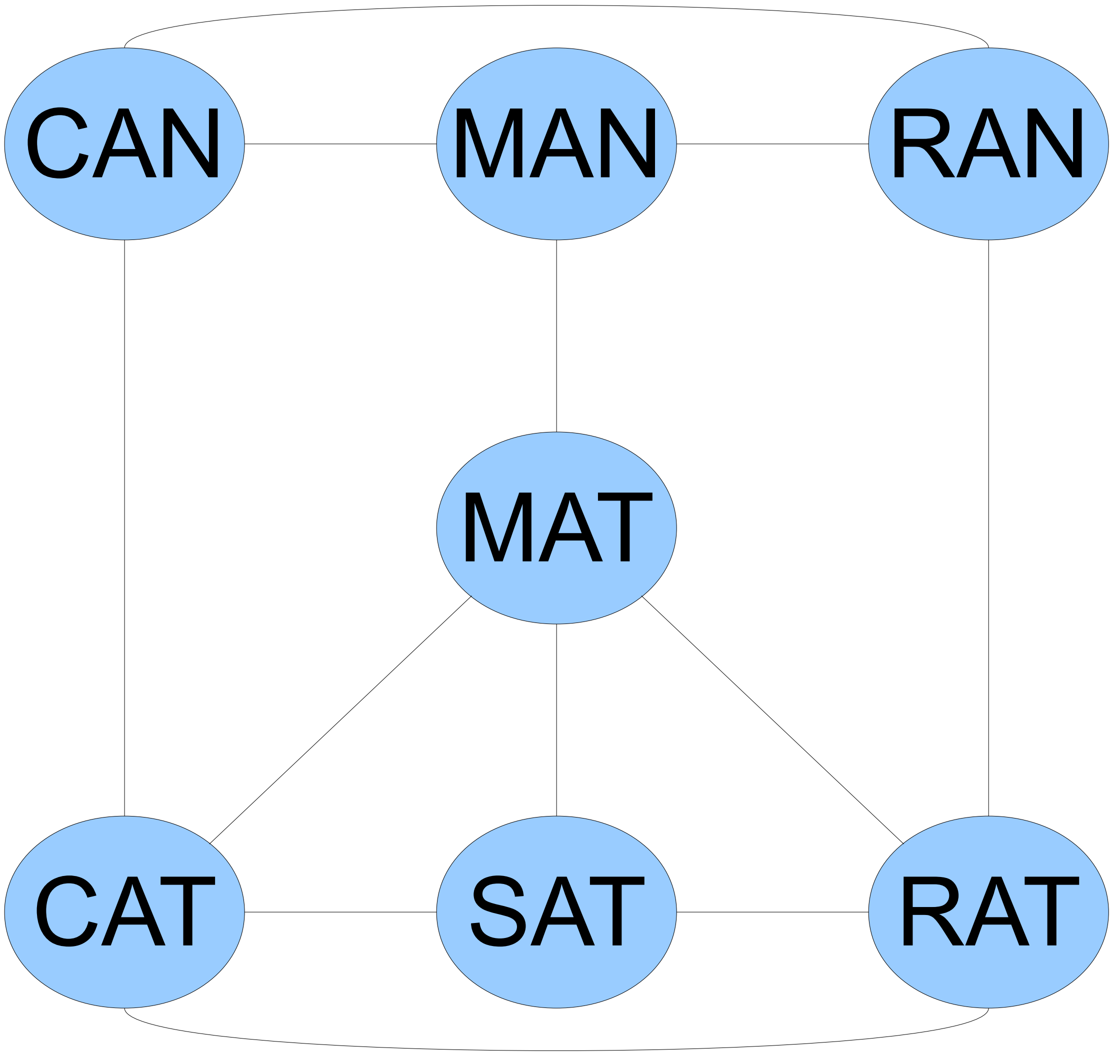


A graph consists of a set of **nodes** connected by **edges**.

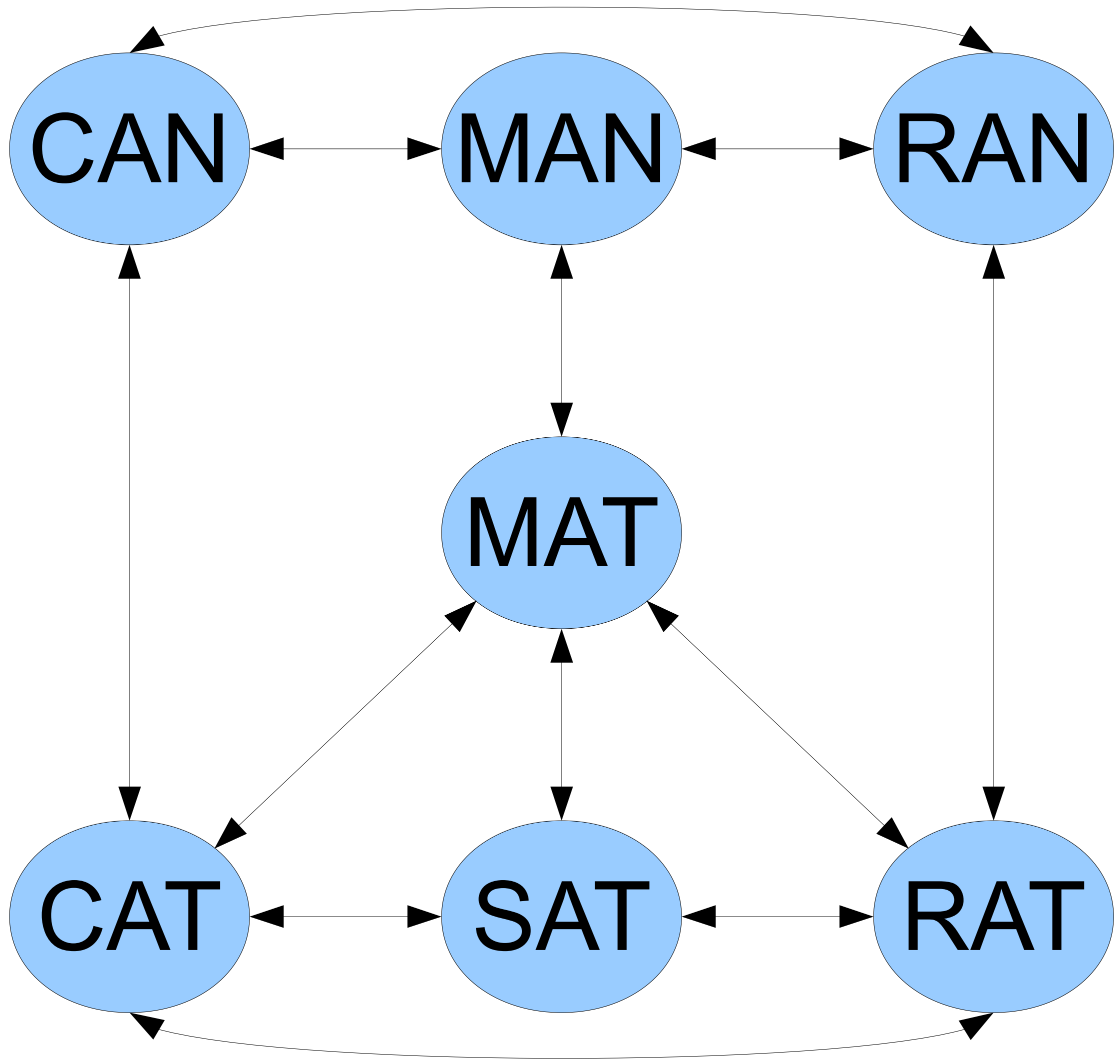
Some graphs are **directed**.



Some graphs are **undirected**.



Some graphs are **undirected**.



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

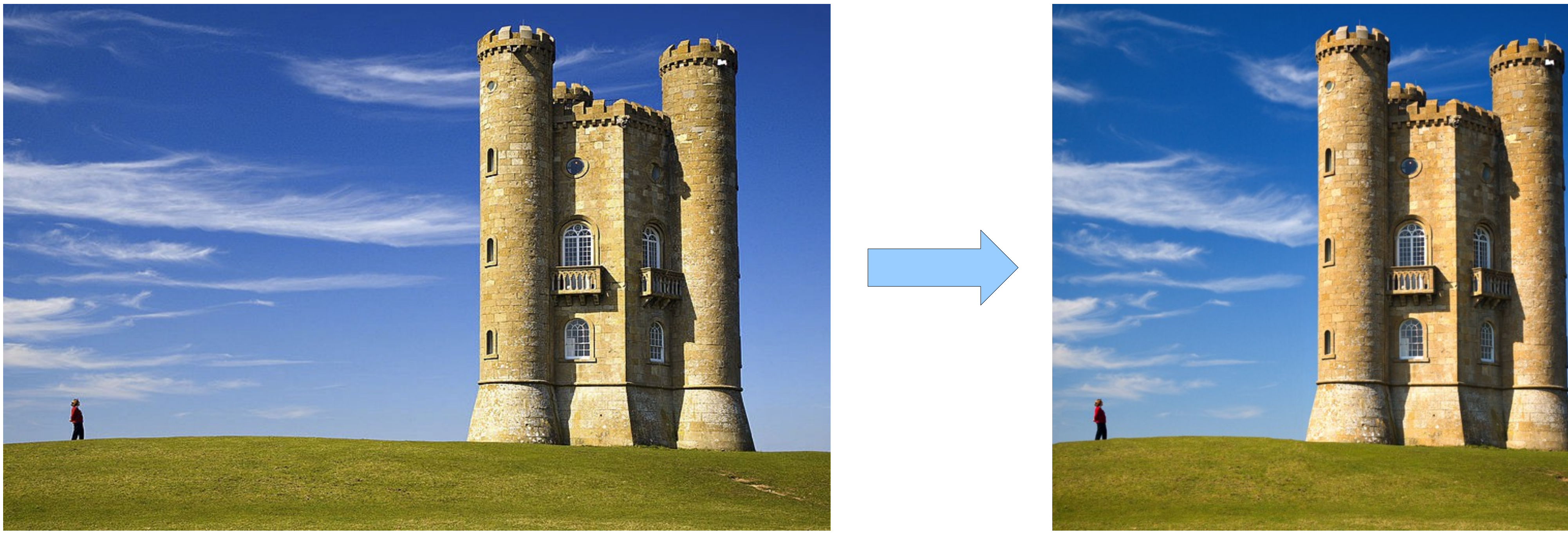
Graphs

- “Yo Teach, why are we studying graphs?”
  - We study graphs because a lot of problems can be modeled in terms of graphs
  - Also, there are many off-the-shelf graph algorithms that we apply if we're able to formulate a problem as a graph problem.

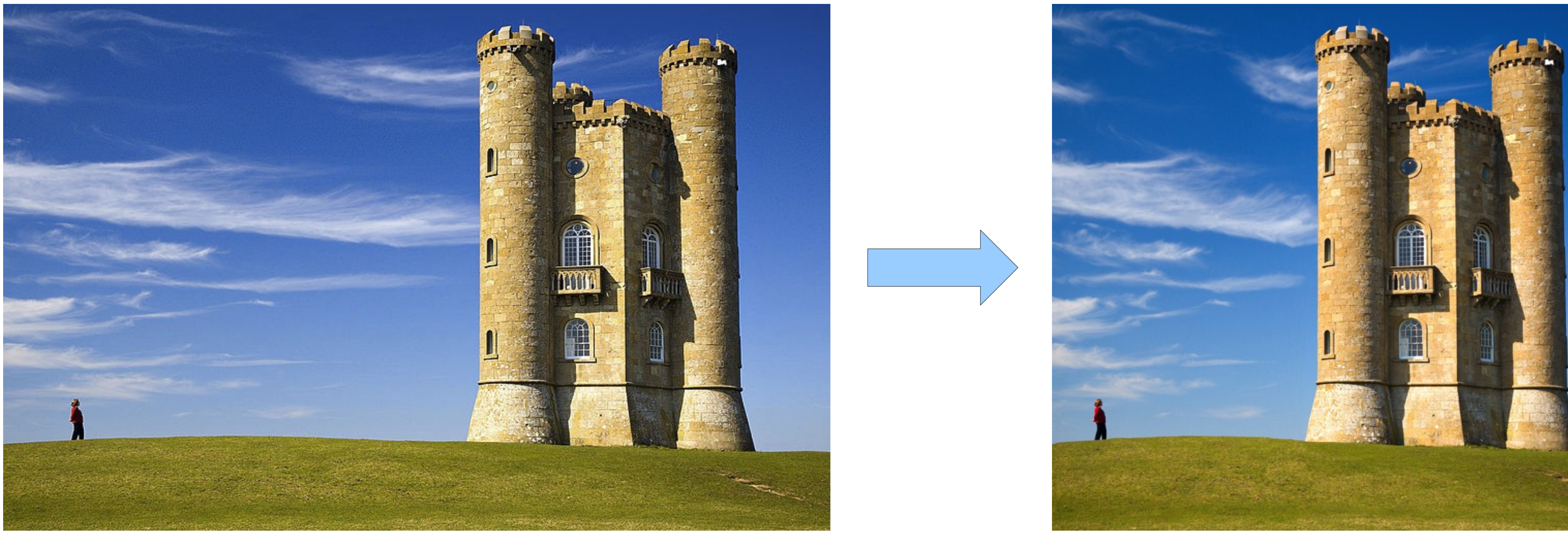
Pathfinding

- Each intersection is a node
- Each street connecting intersections is an edge
- Find paths between intersections that minimize distance or travel time

Content-Aware Resizing



Content-Aware Resizing

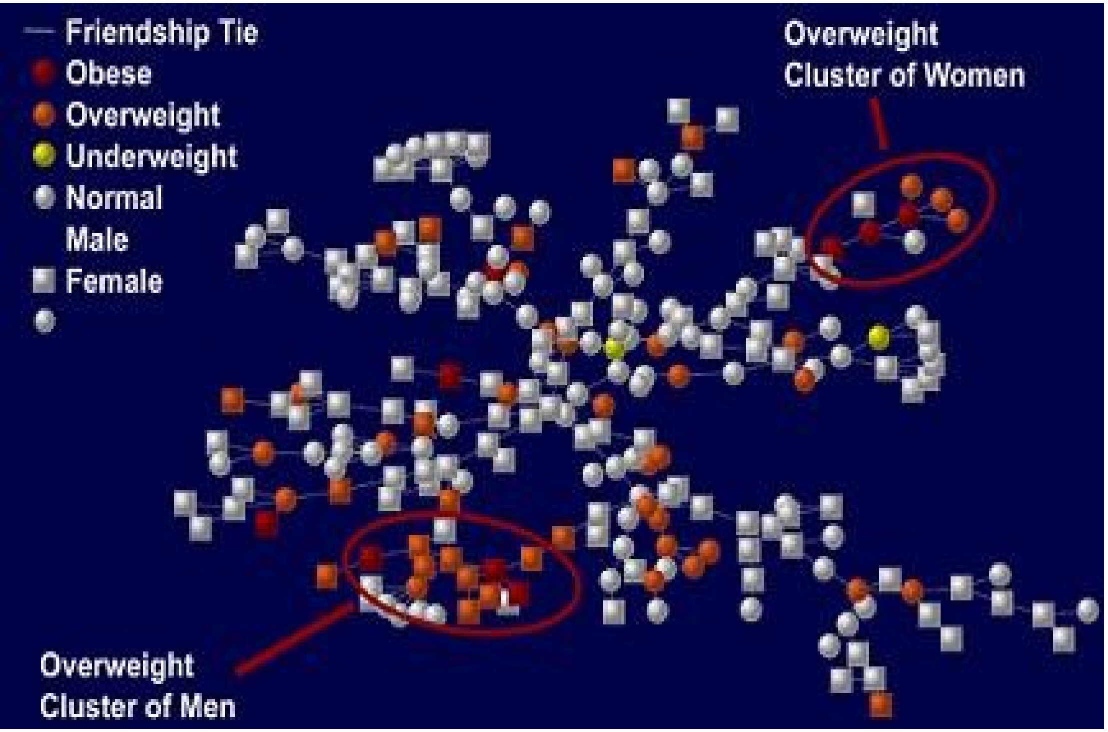


- Each pixel is a node in a graph
- Each pixel is connected to adjacent pixels
- Find paths from the top of the image to the bottom that minimize the “energy function”

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.

Social Networks and Epidemics



http://3278as3udzze1hdk0f2th5nf18c1.wpengine.netdna-cdn.com/wp-content/uploads/2010/09/social-networks-new-science1.jpg

https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&docid=MPB6JuYe9sdbDM&tbnid=VWbN0BtTG7-ZWM:&ved=0CAUQjRw&url=http%3A%2F%2Fblogs.cornell.edu%2Finfo2040%2F2011%2F09%2F17%2Fhow-network-structure-can-provide-advanced-warning-about-epidemics%2F&ei=rrP9Ufa8G6n0iwK9q4CoBQ&bvm=bv.50165853,d.cGE&psig=AFQjCNHljXC1ZggH6hQDrKxPM1CV7ocVXQ&ust=1375667391653650

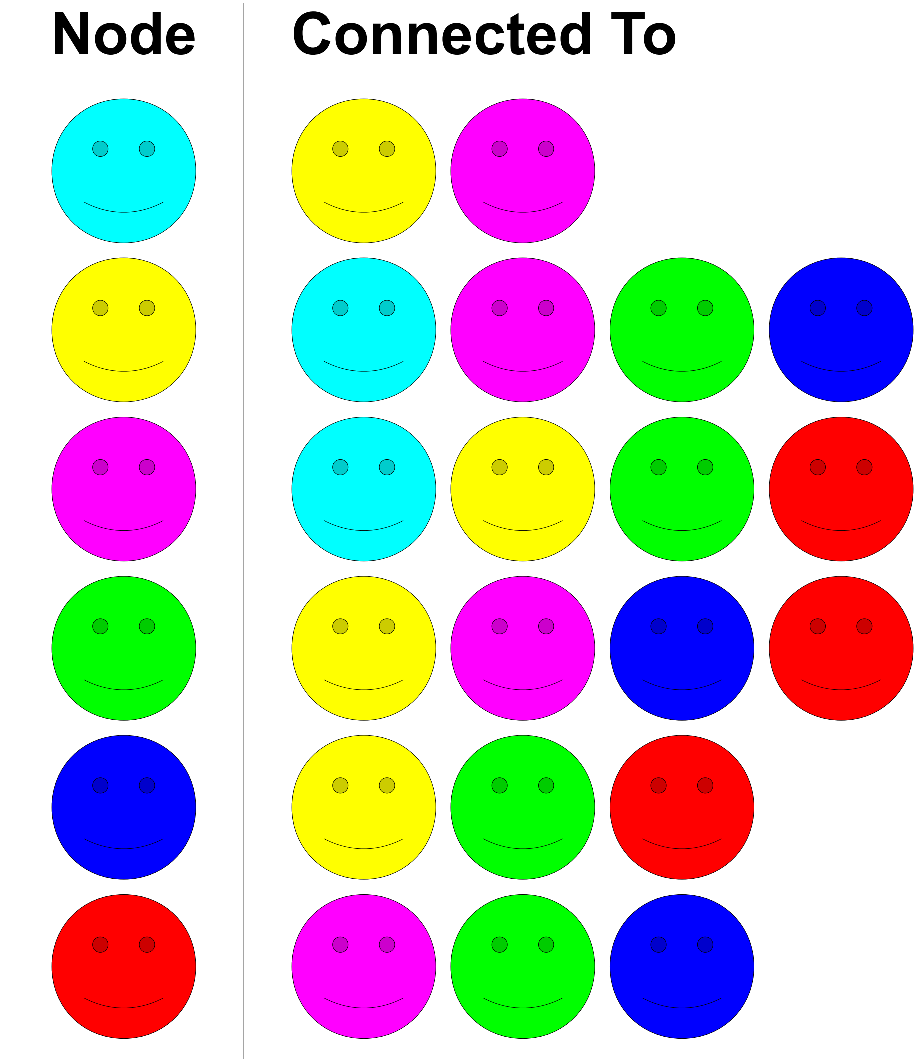
How can we represent graphs in C++?

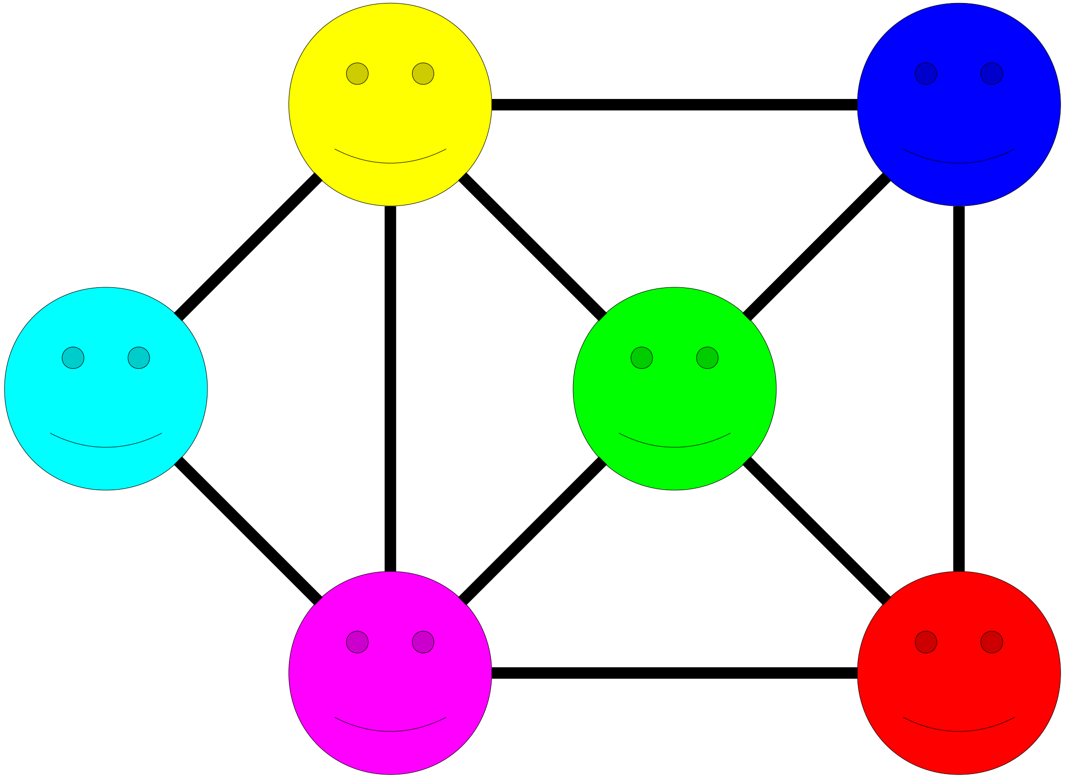
Representing Graphs

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*>`



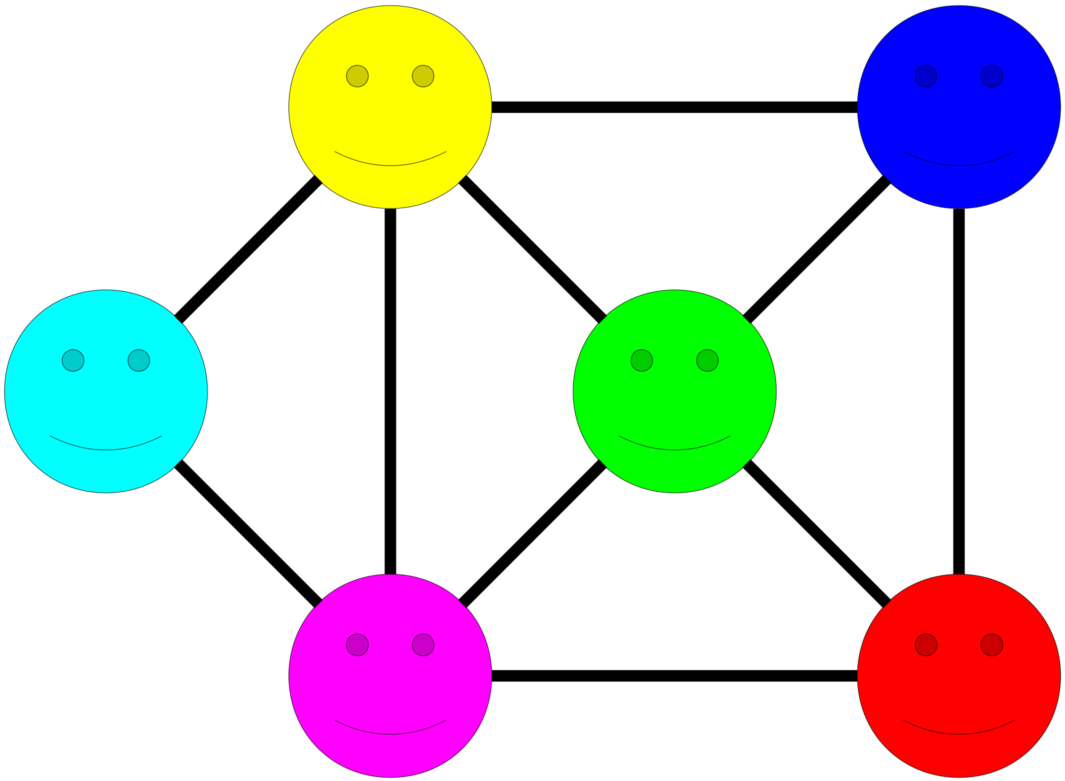


What interesting things can we do with graphs?

Connected Components

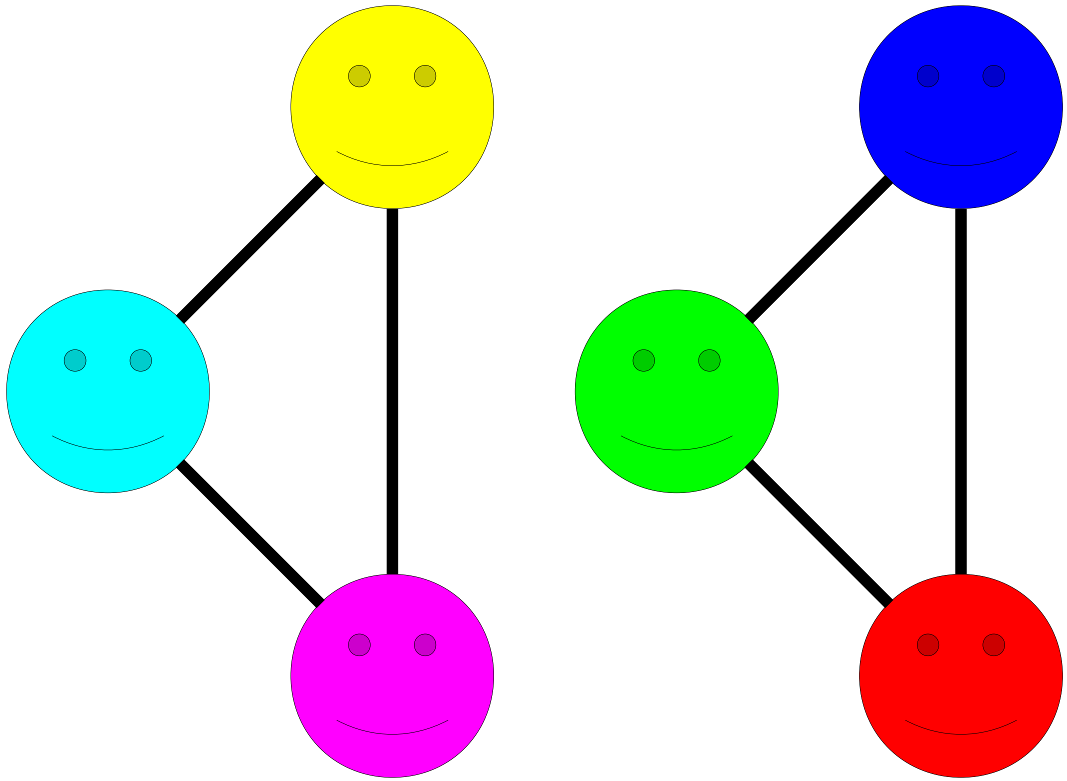
- A **connected component** is a subset of the nodes in a graph such that:
  - For every pair of nodes in the subset there exists a path between them
  - No node in the subset is not connected any node not in the subset

Connected Components



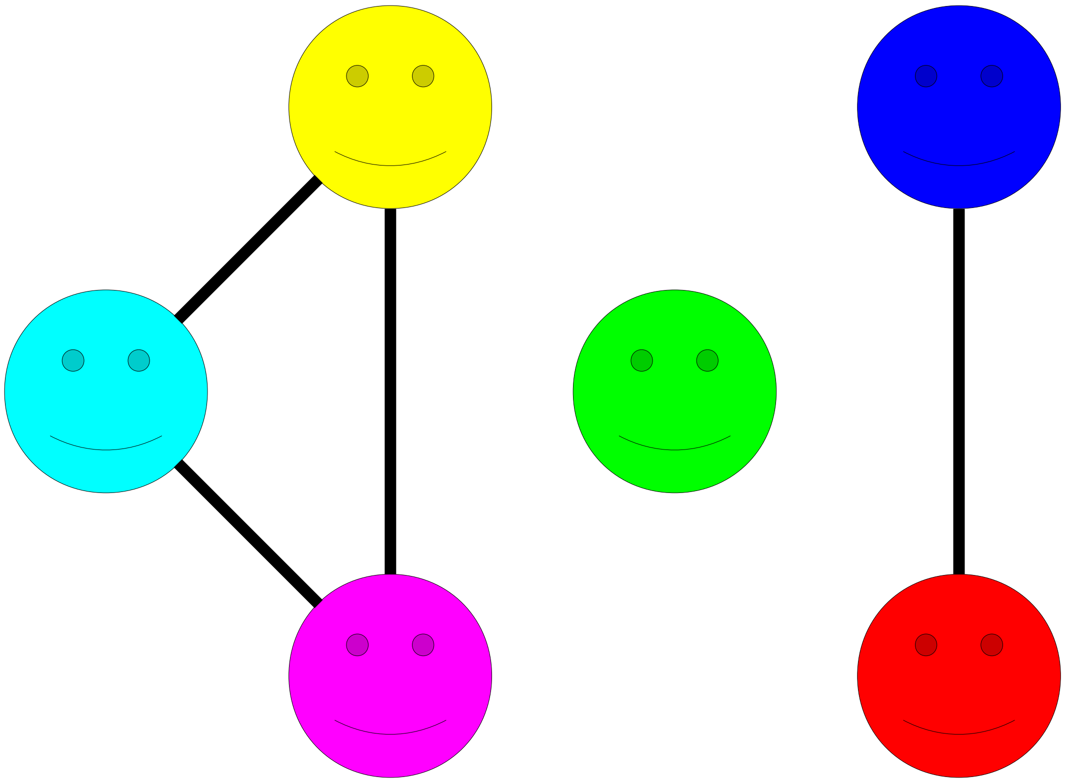
1 Connected Component

Connected Components



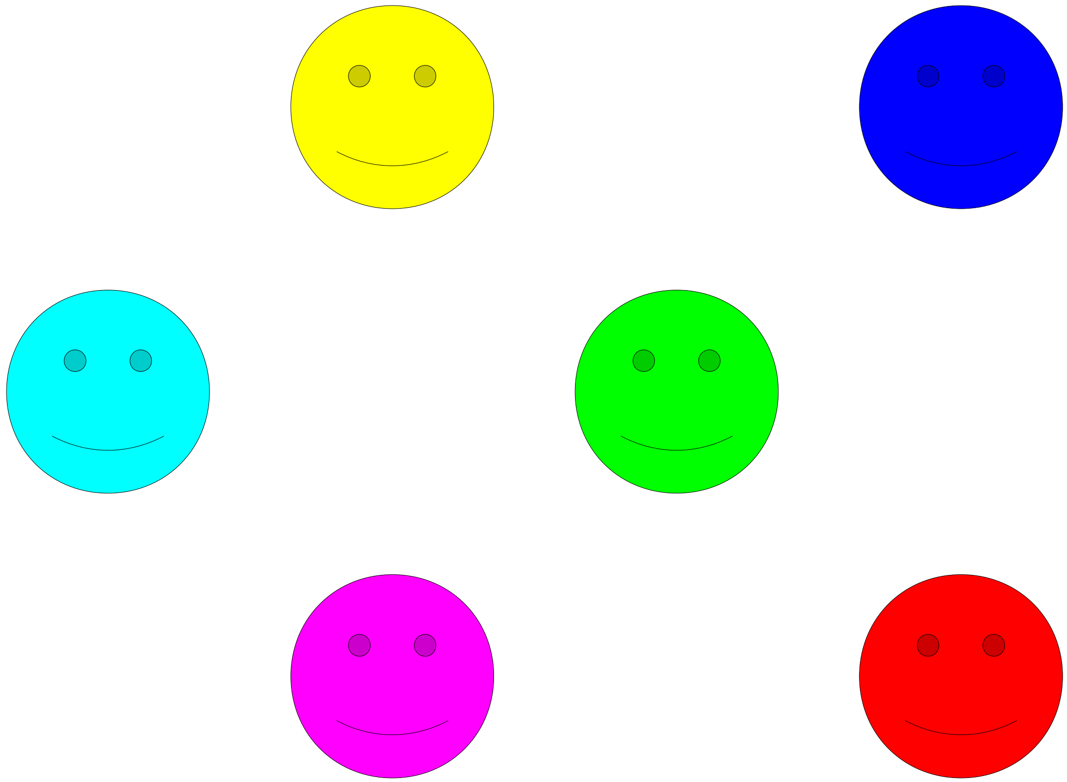
2 Connected Components

Connected Components



3 Connected Components

Connected Components



6 Connected Components

Connected Components

- Detecting connected components in a graph is important because it can provide useful insights into the structure of graph
  - e.g. How do people in a community separate themselves into separate groups.
- In order to detect connected components we first need to be able to iterate over nodes in a graph.
- We'll come back to connected components later.

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.

Iterating over a Graph

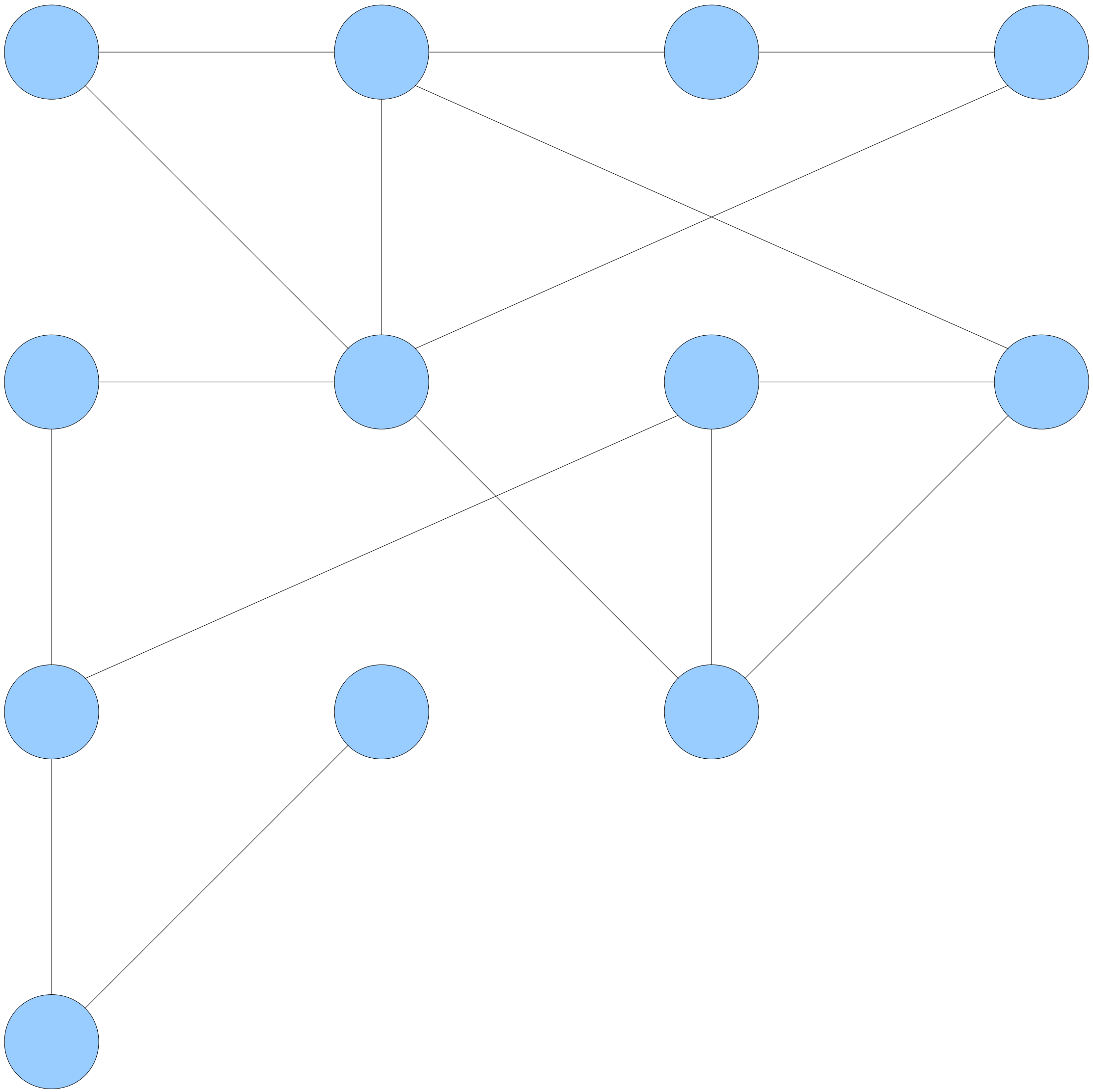
- All methods of iterating over a graph involve keeping track of 3 sets of nodes:

Set of Nodes already visited

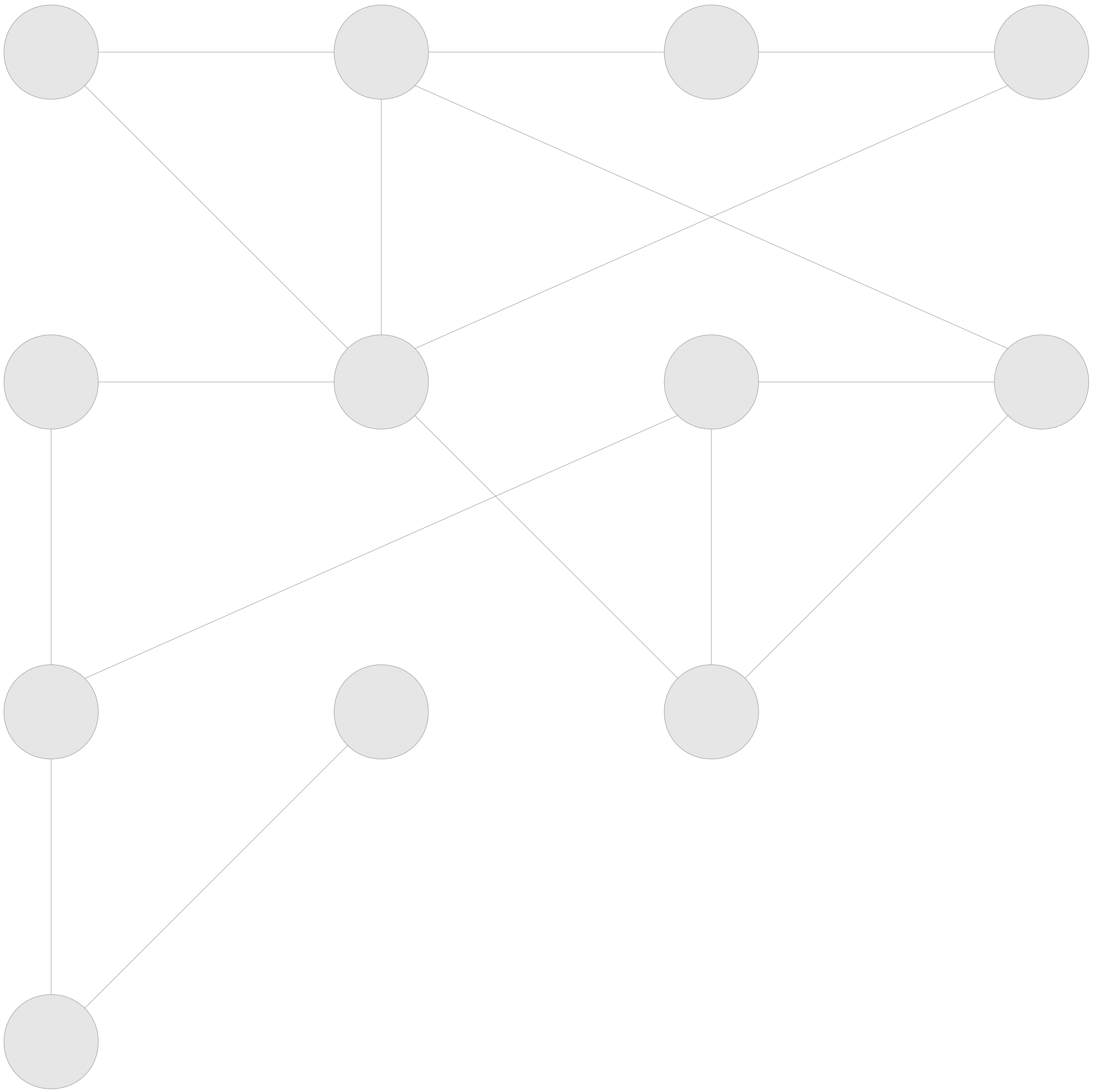
Set of Nodes to look at next

Everything else

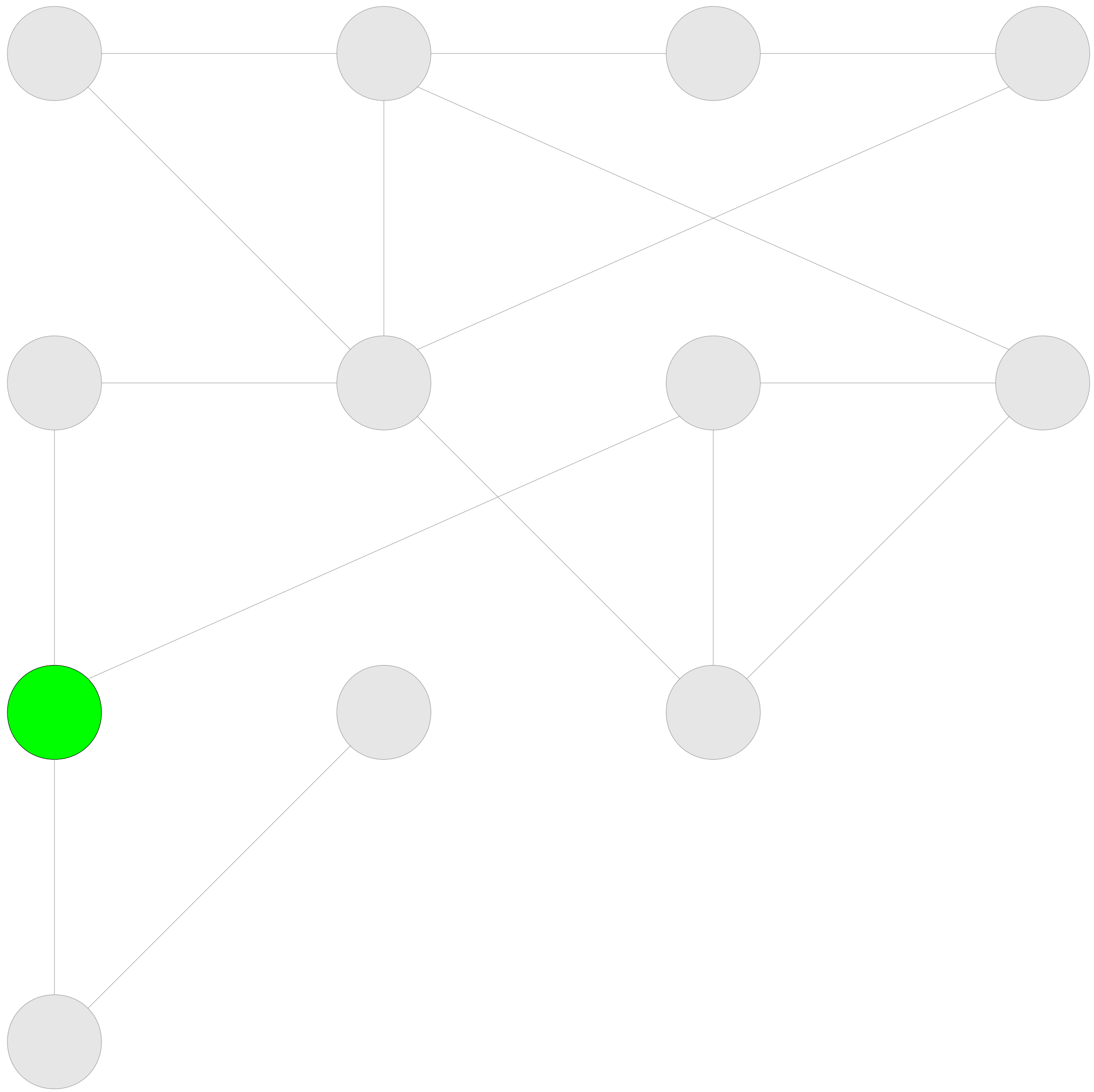
Iterating over a Graph



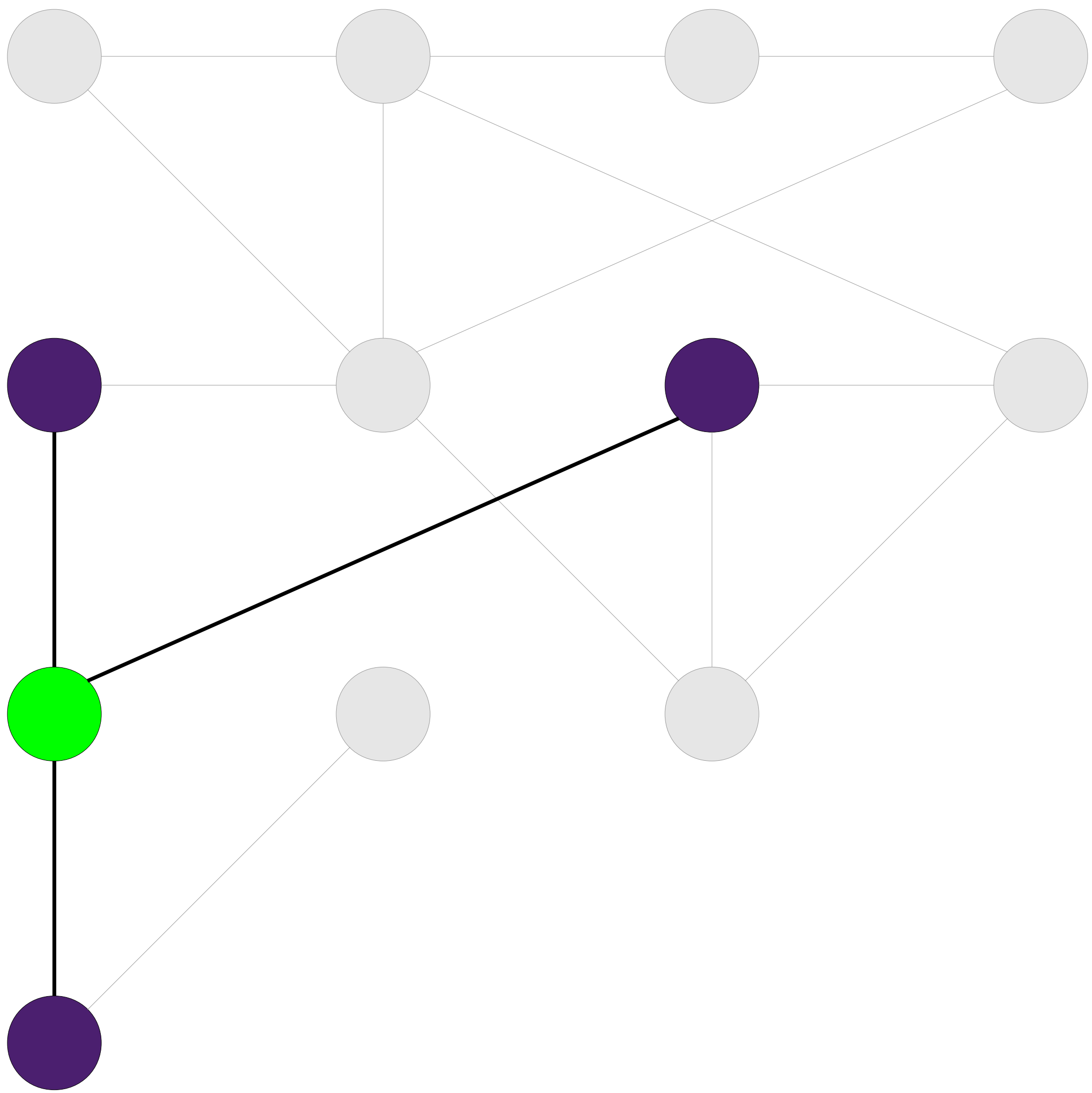
Iterating over a Graph



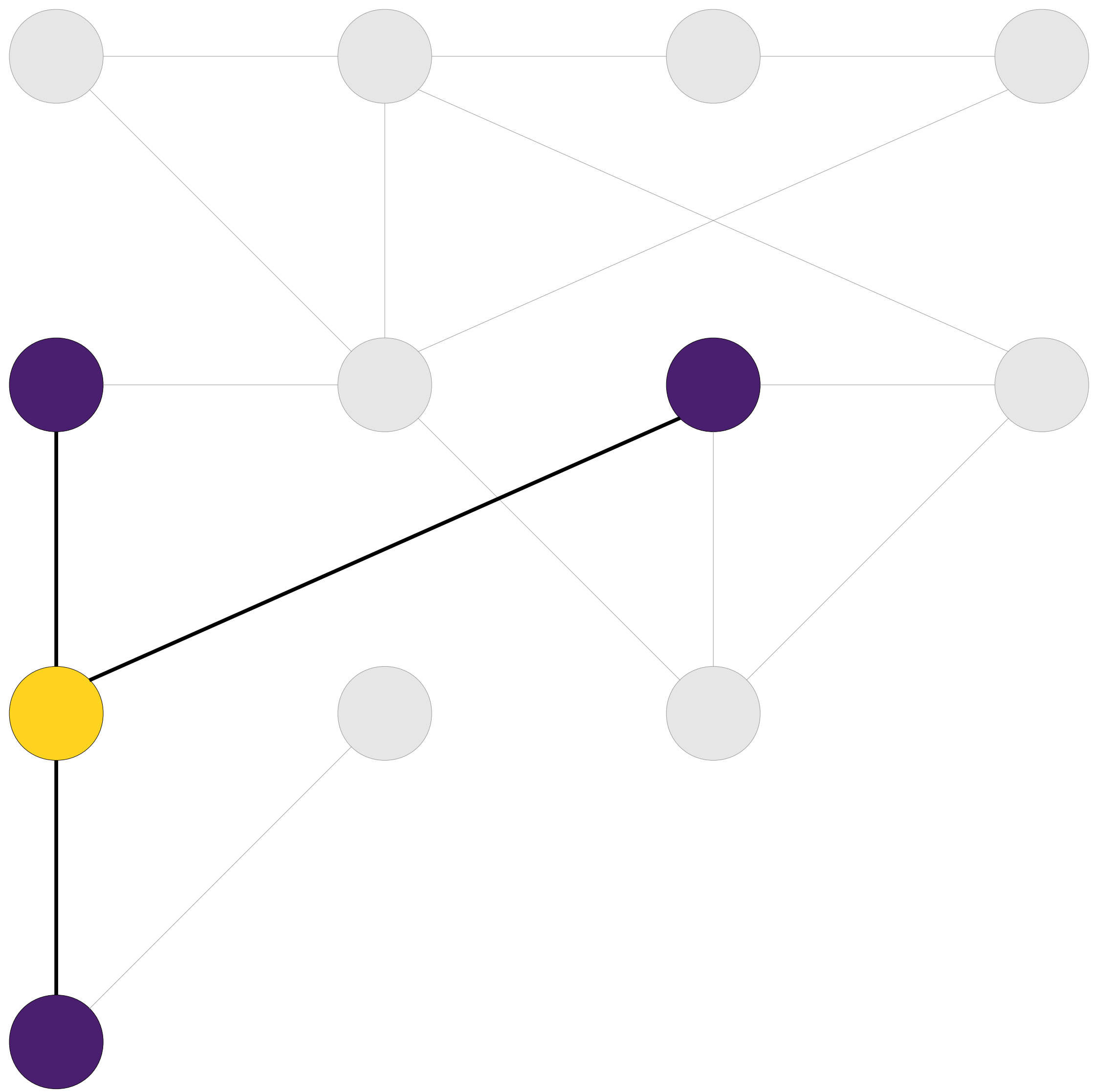
Iterating over a Graph



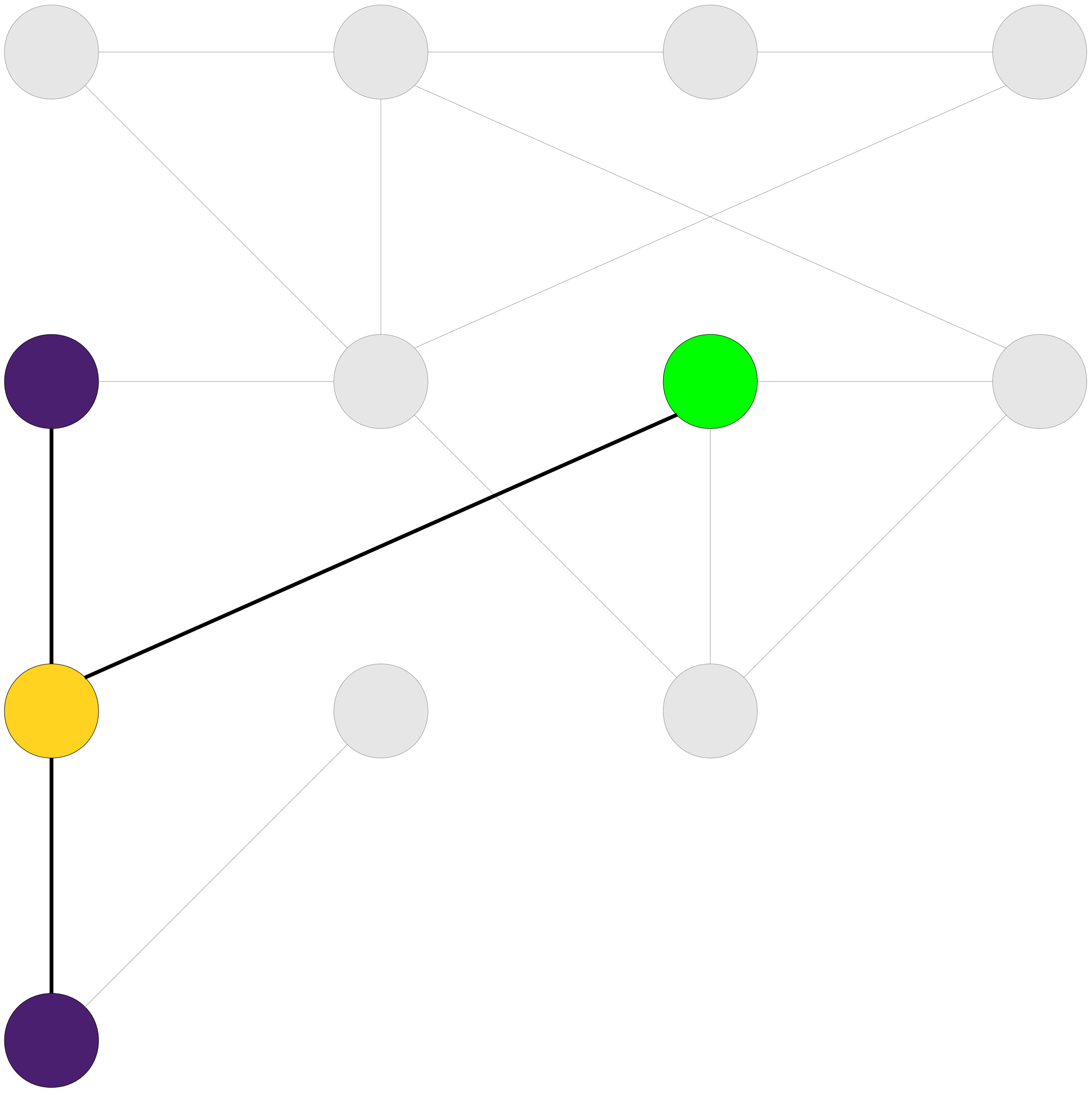
Iterating over a Graph



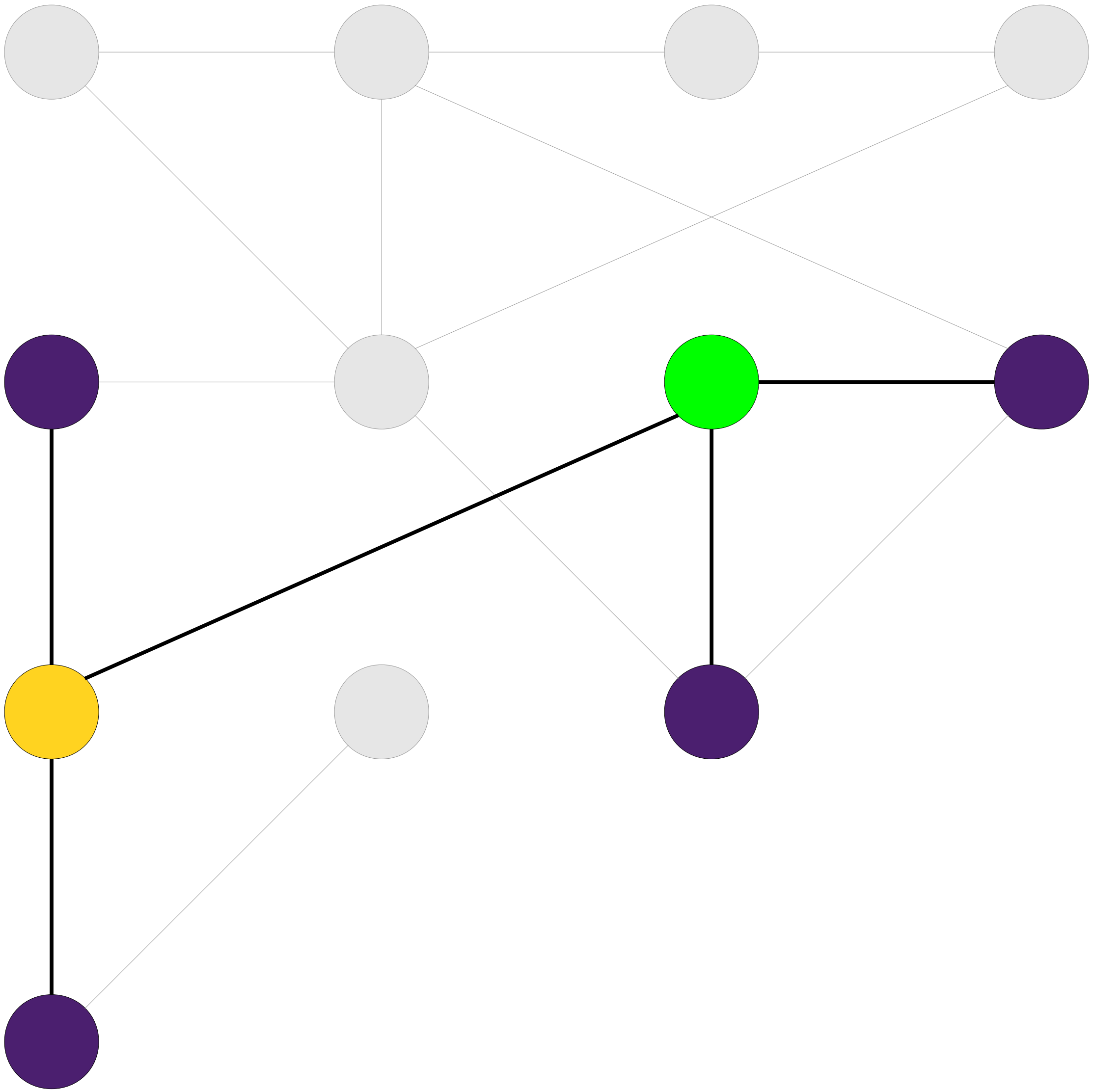
Iterating over a Graph



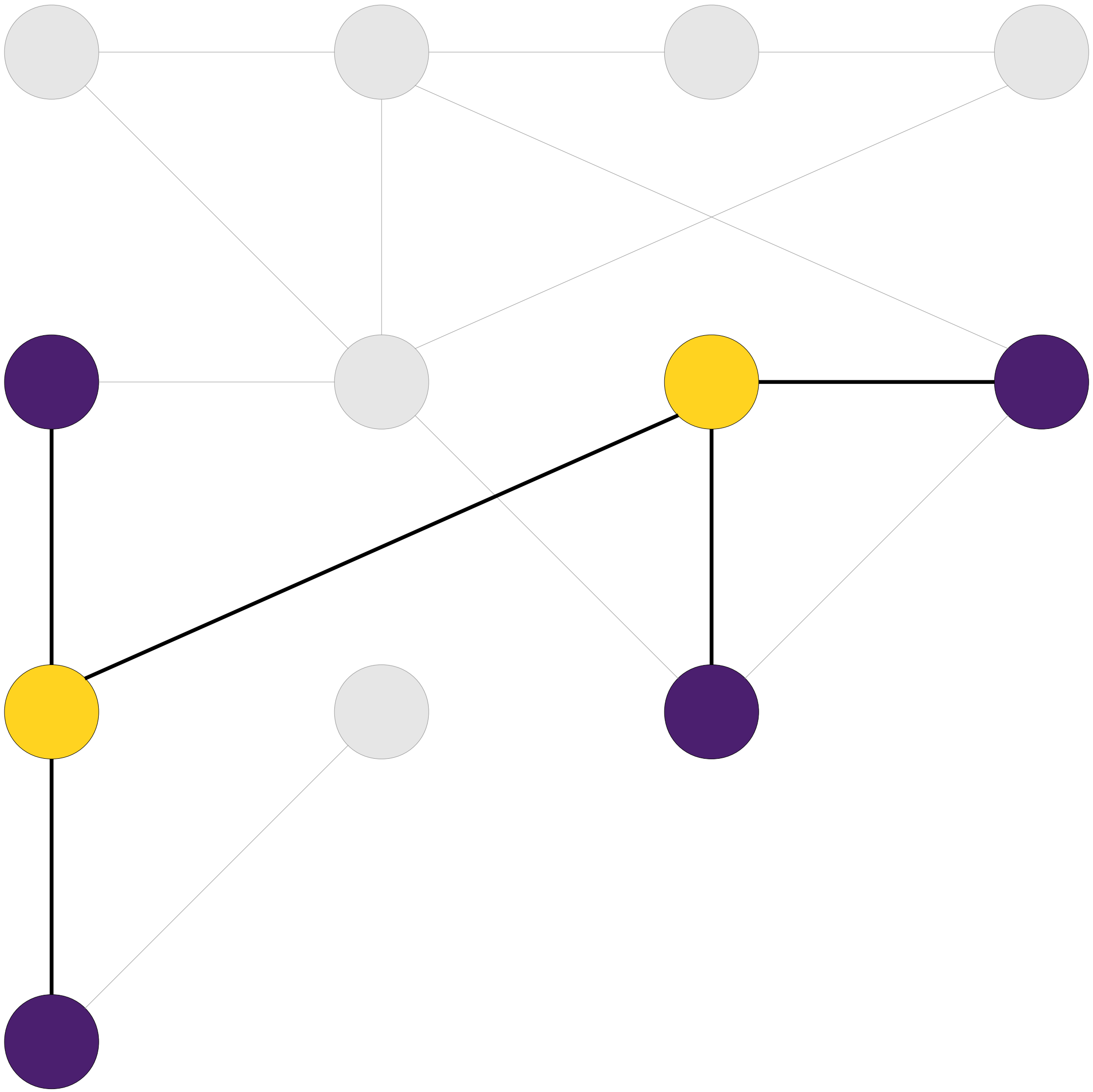
Iterating over a Graph



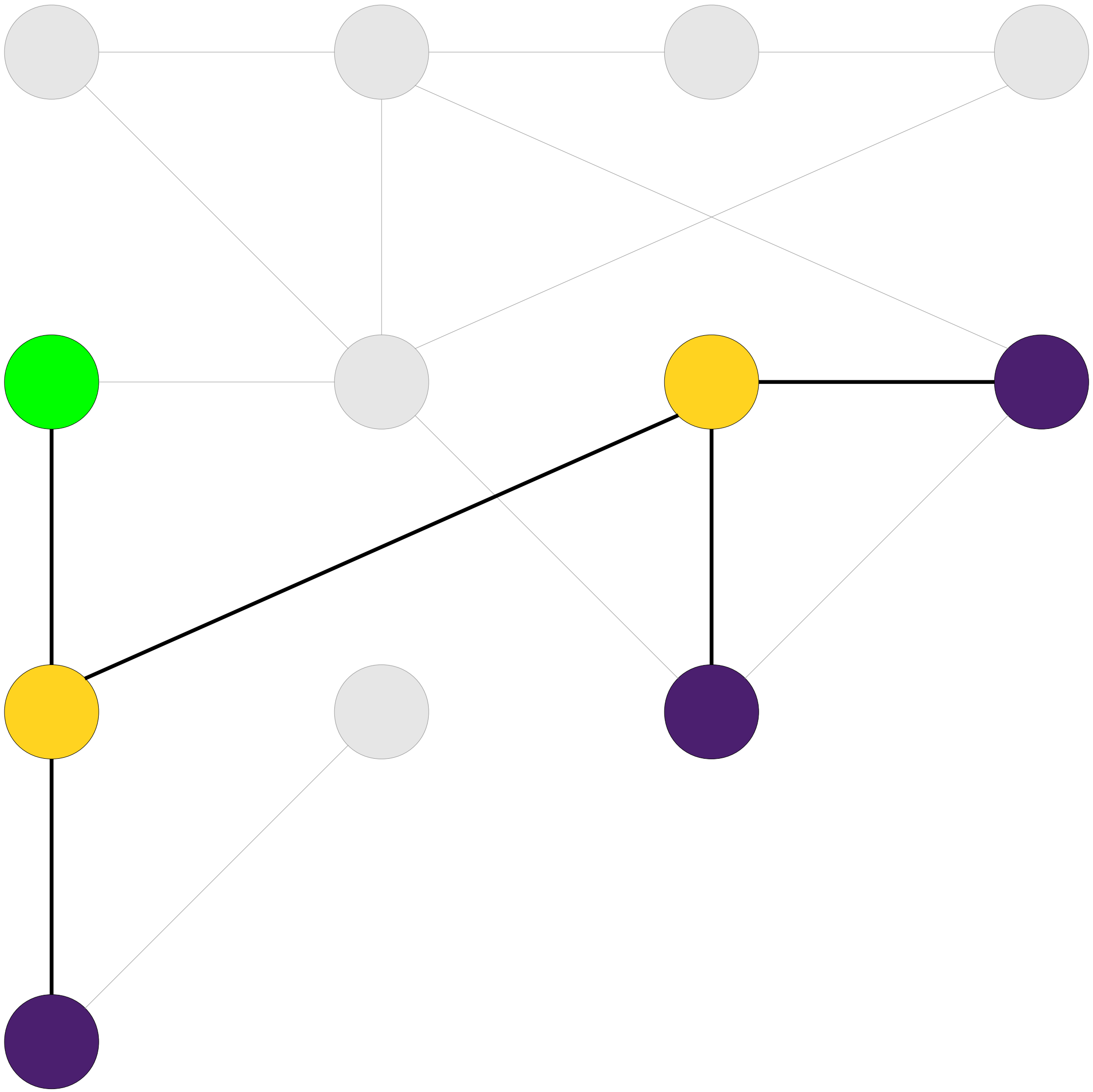
Iterating over a Graph



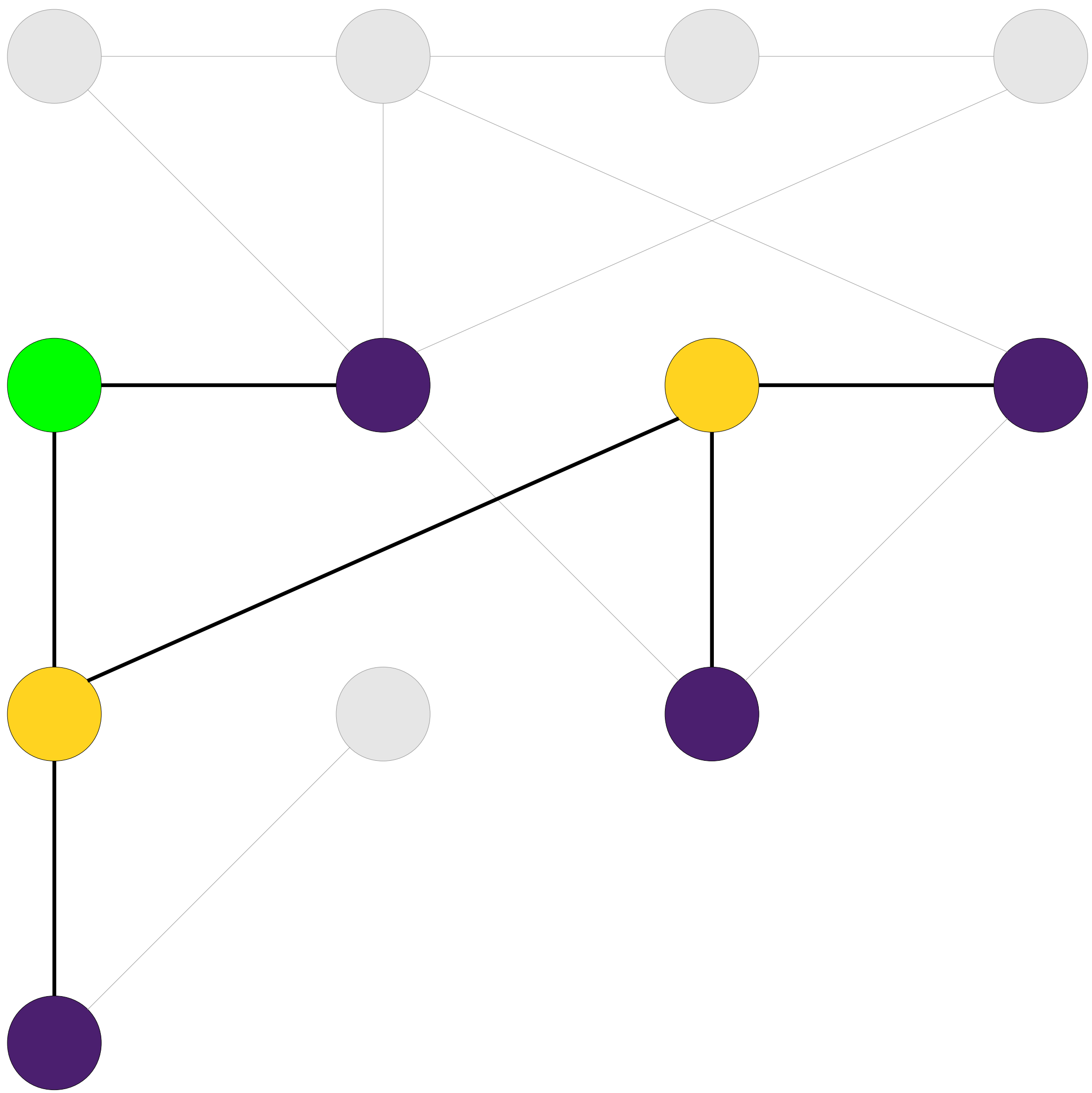
Iterating over a Graph



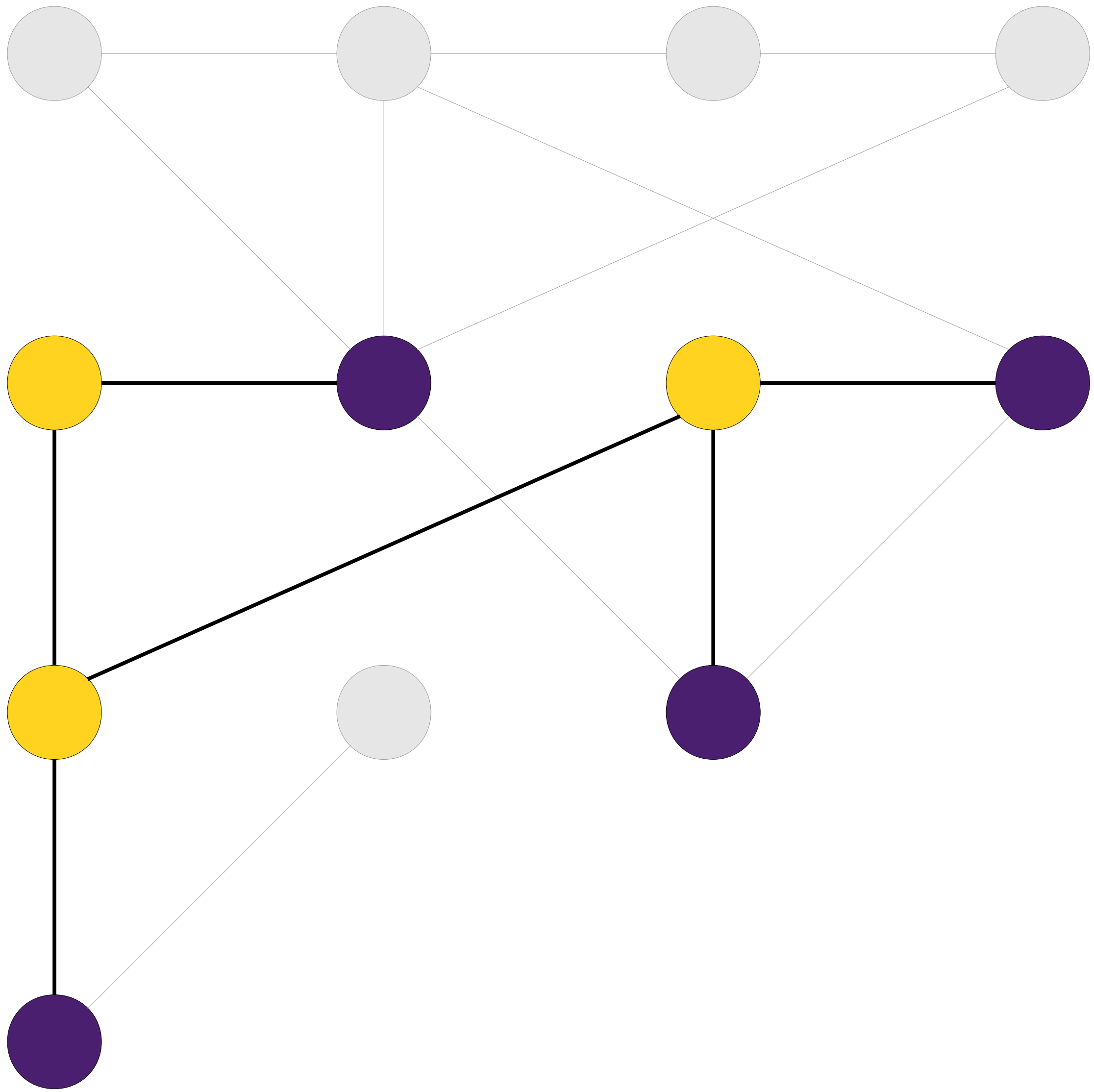
Iterating over a Graph



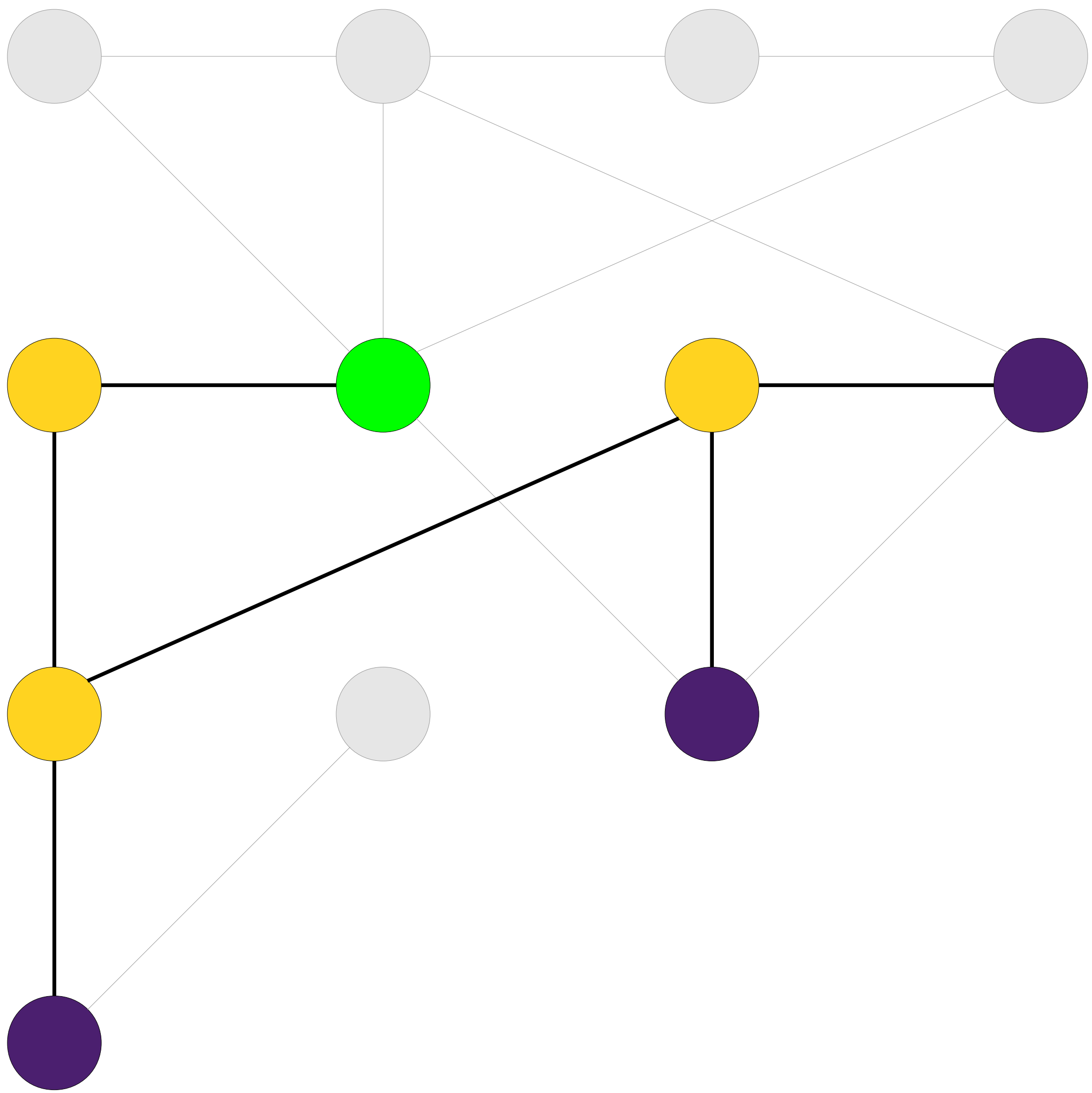
Iterating over a Graph



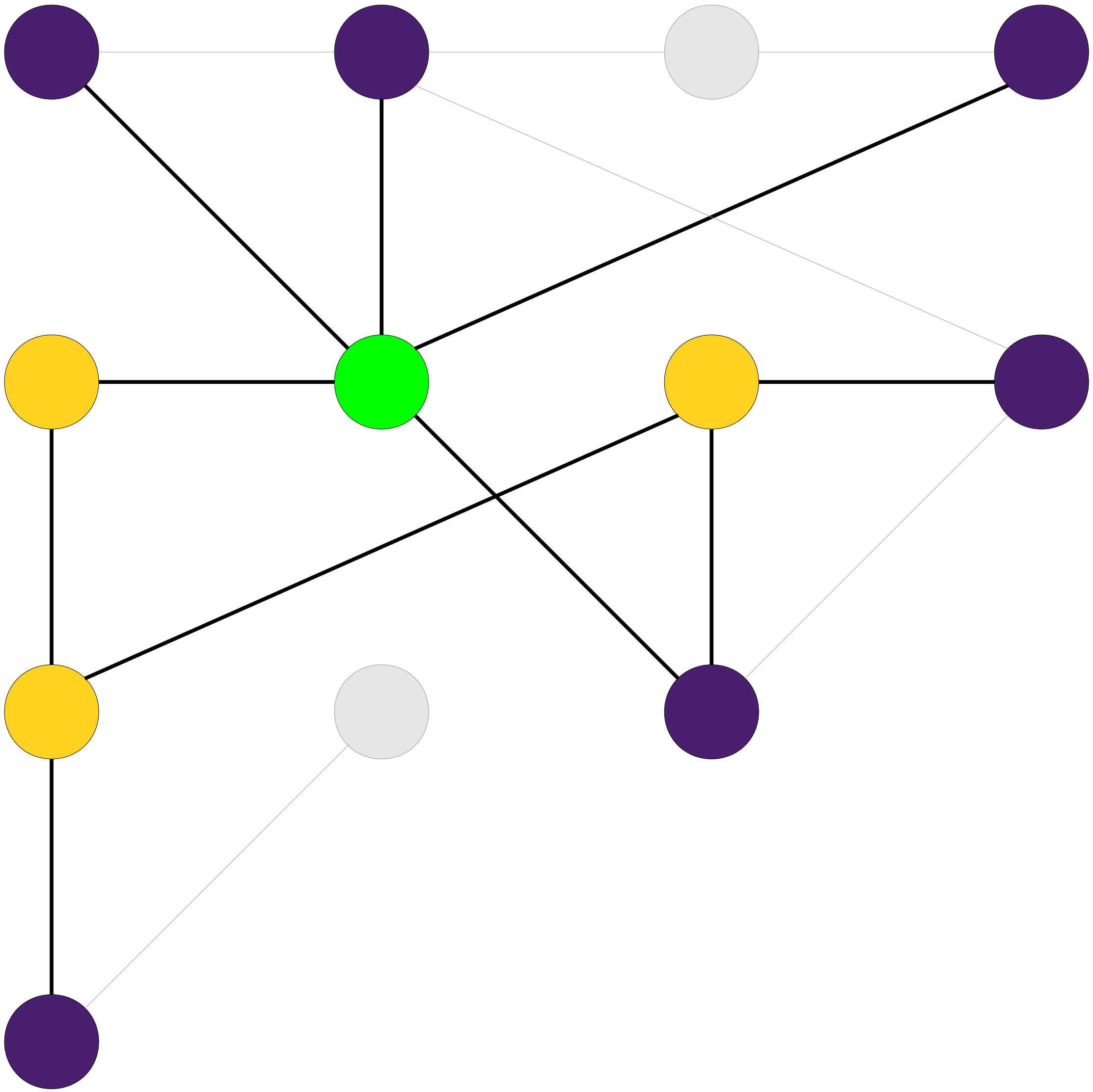
Iterating over a Graph



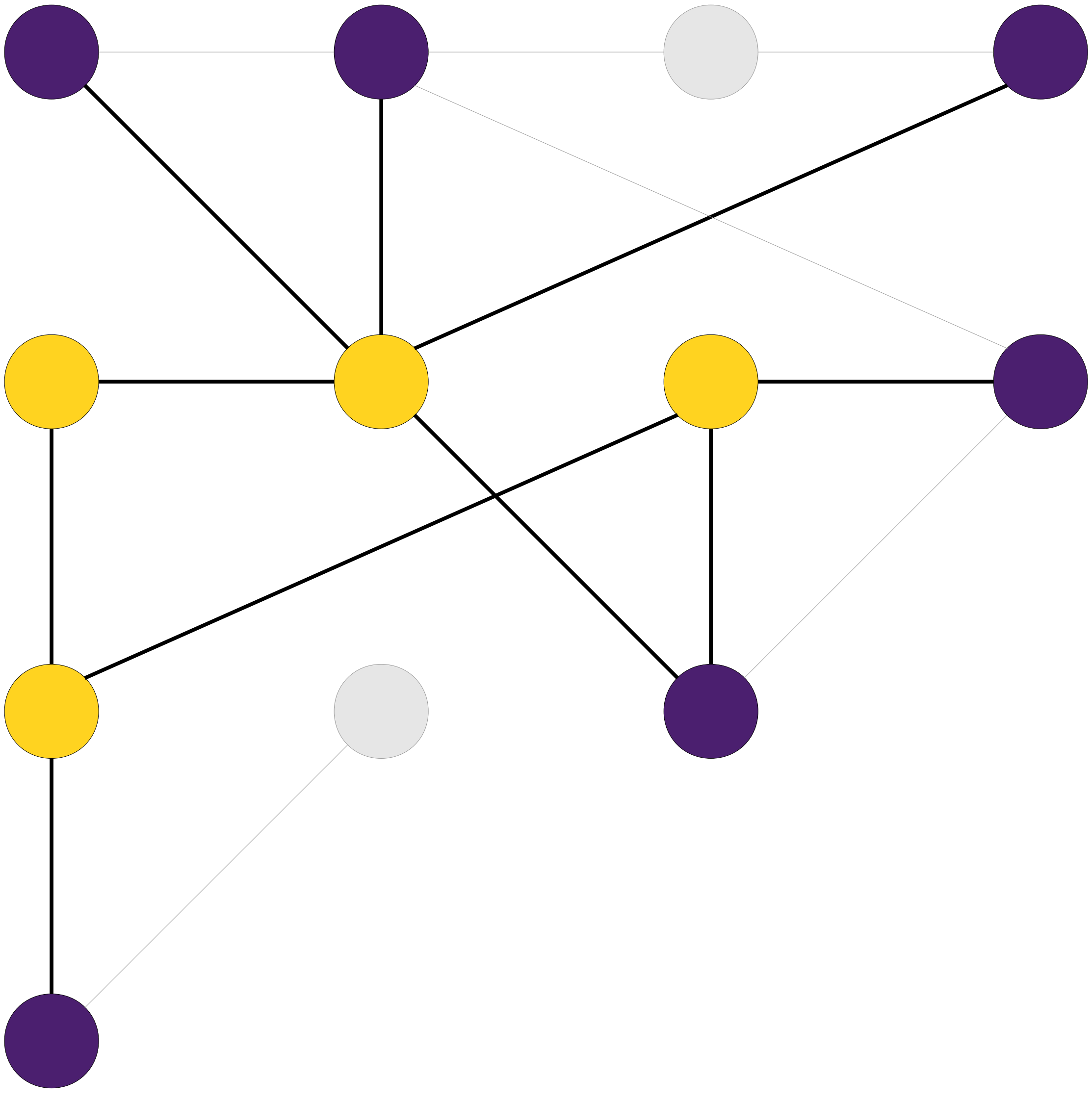
Iterating over a Graph



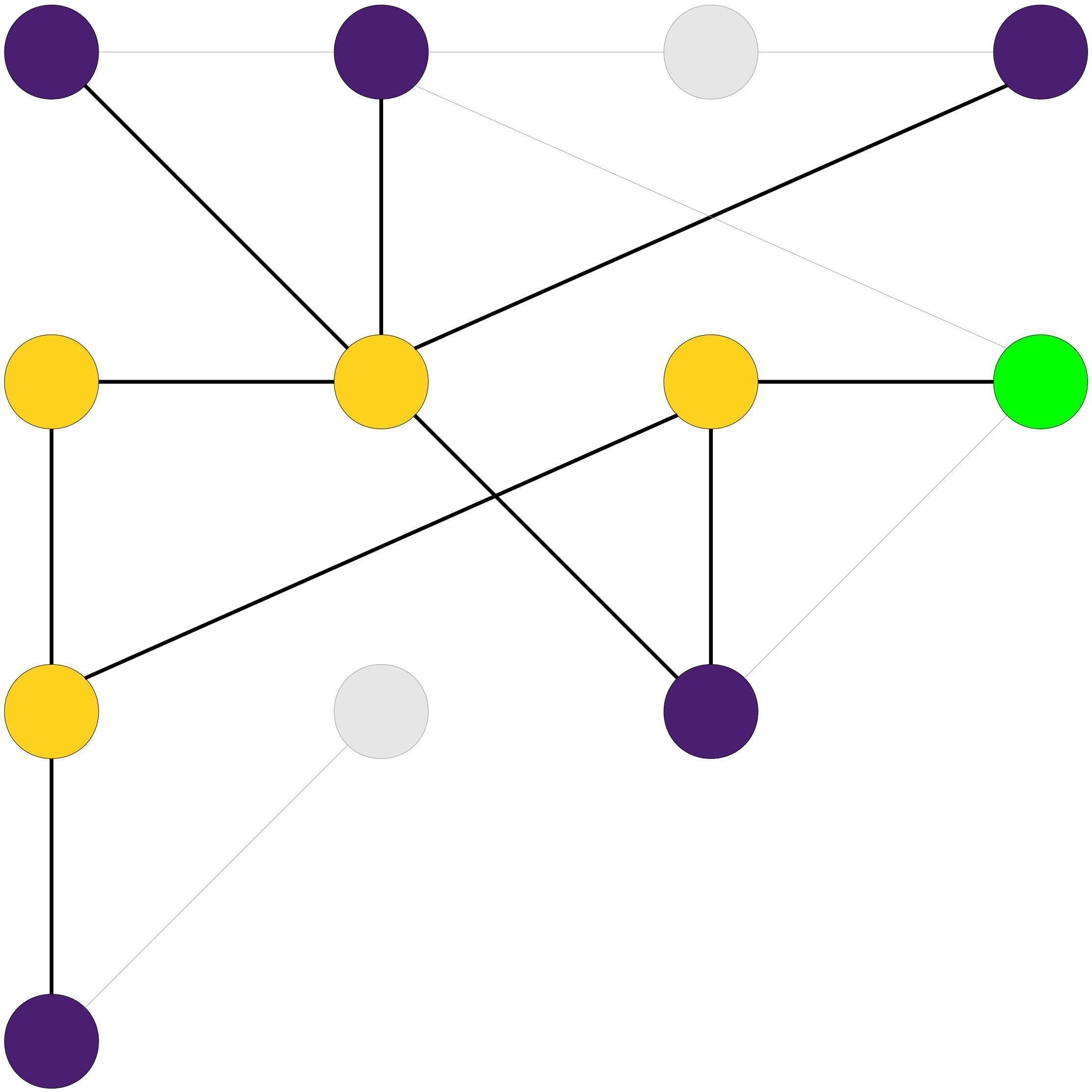
Iterating over a Graph



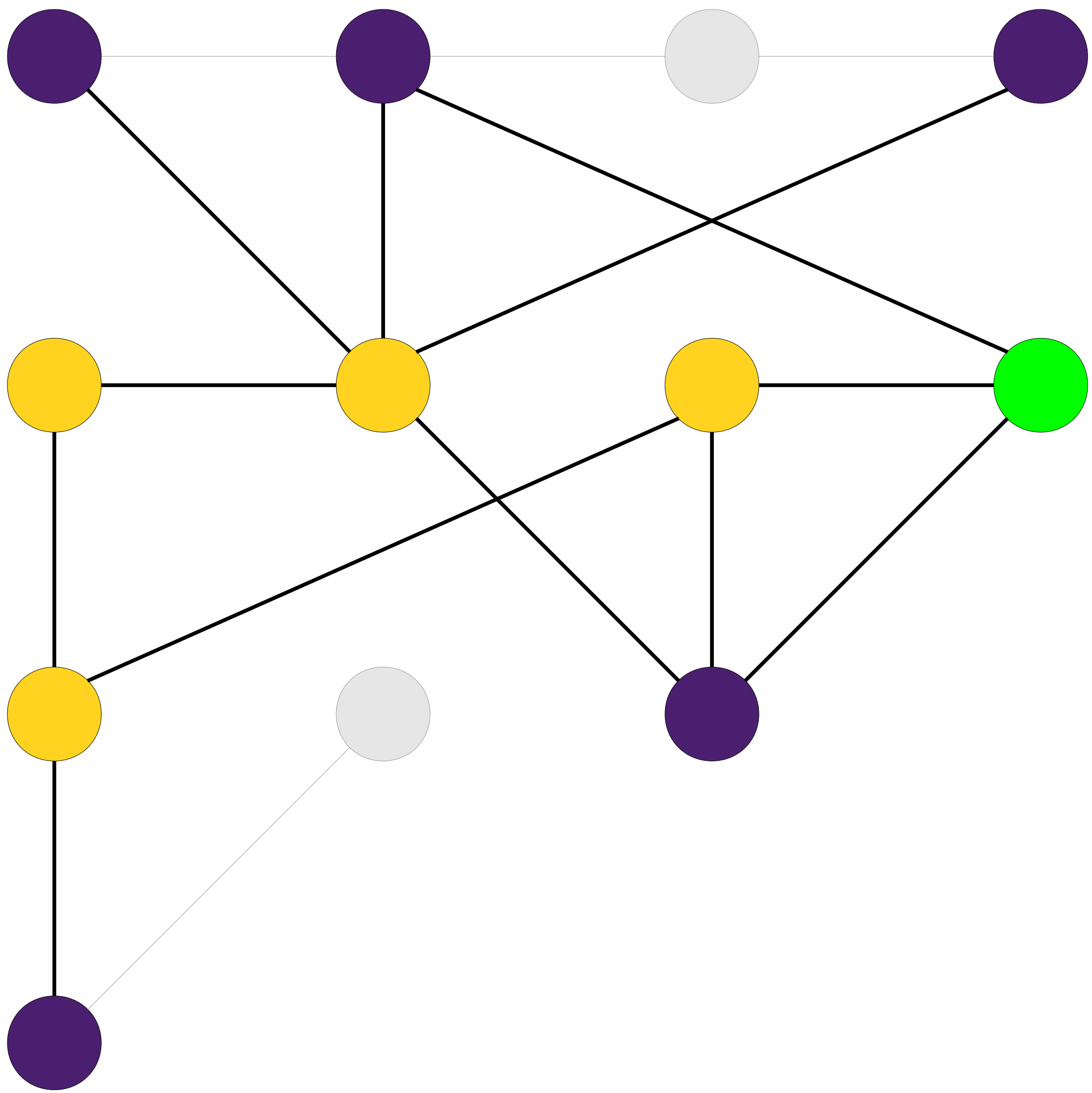
Iterating over a Graph



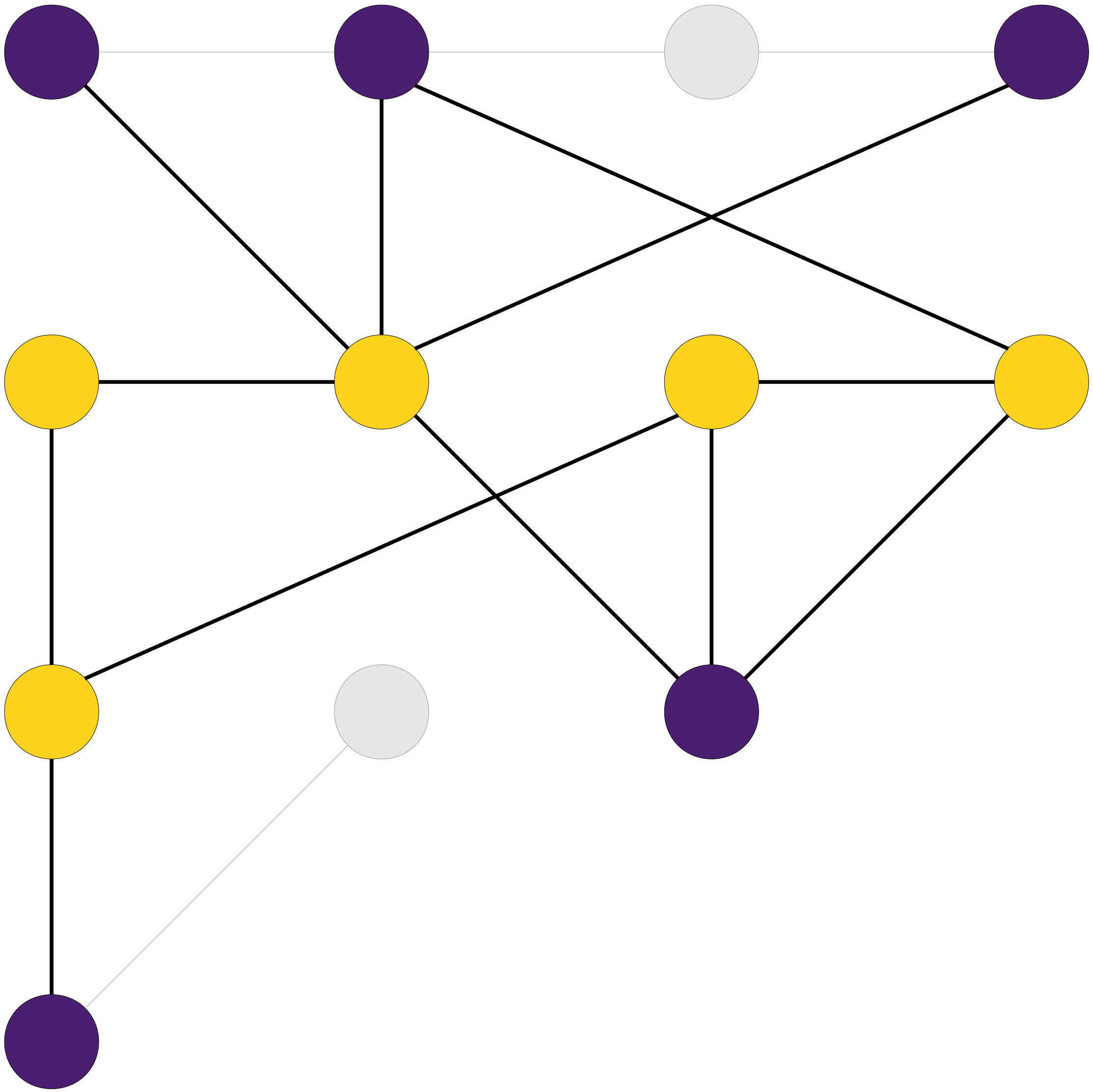
Iterating over a Graph



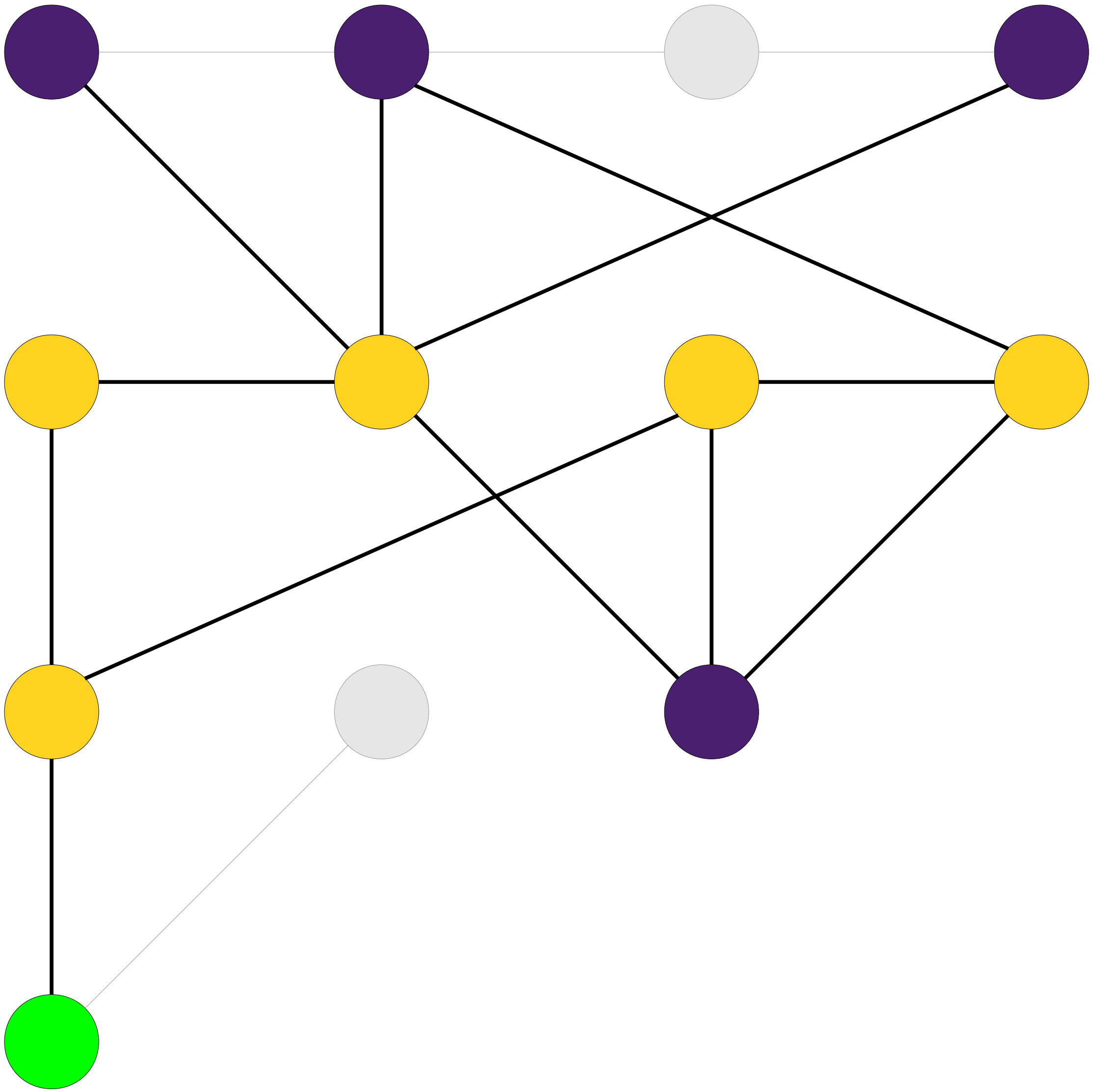
Iterating over a Graph



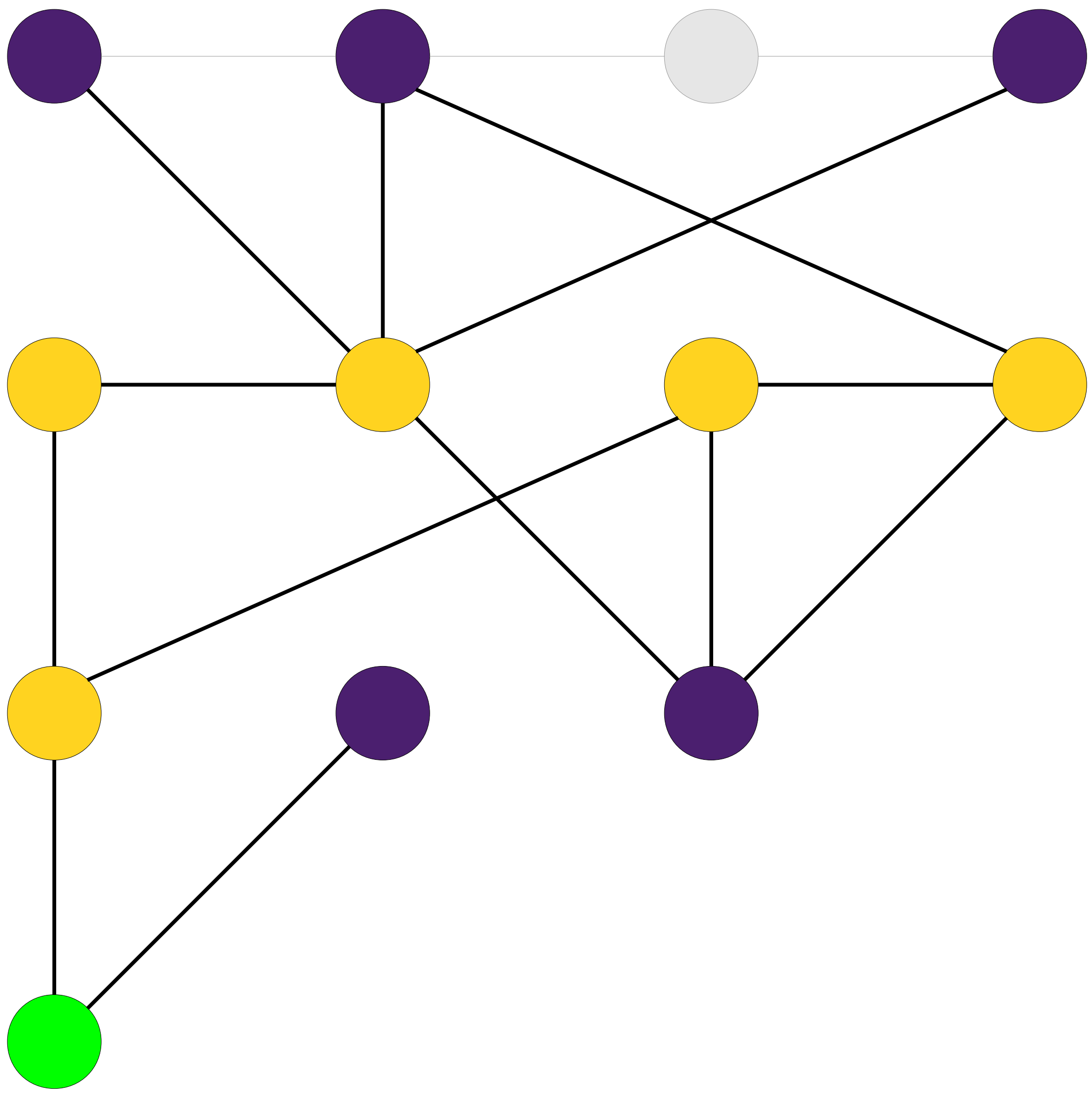
Iterating over a Graph



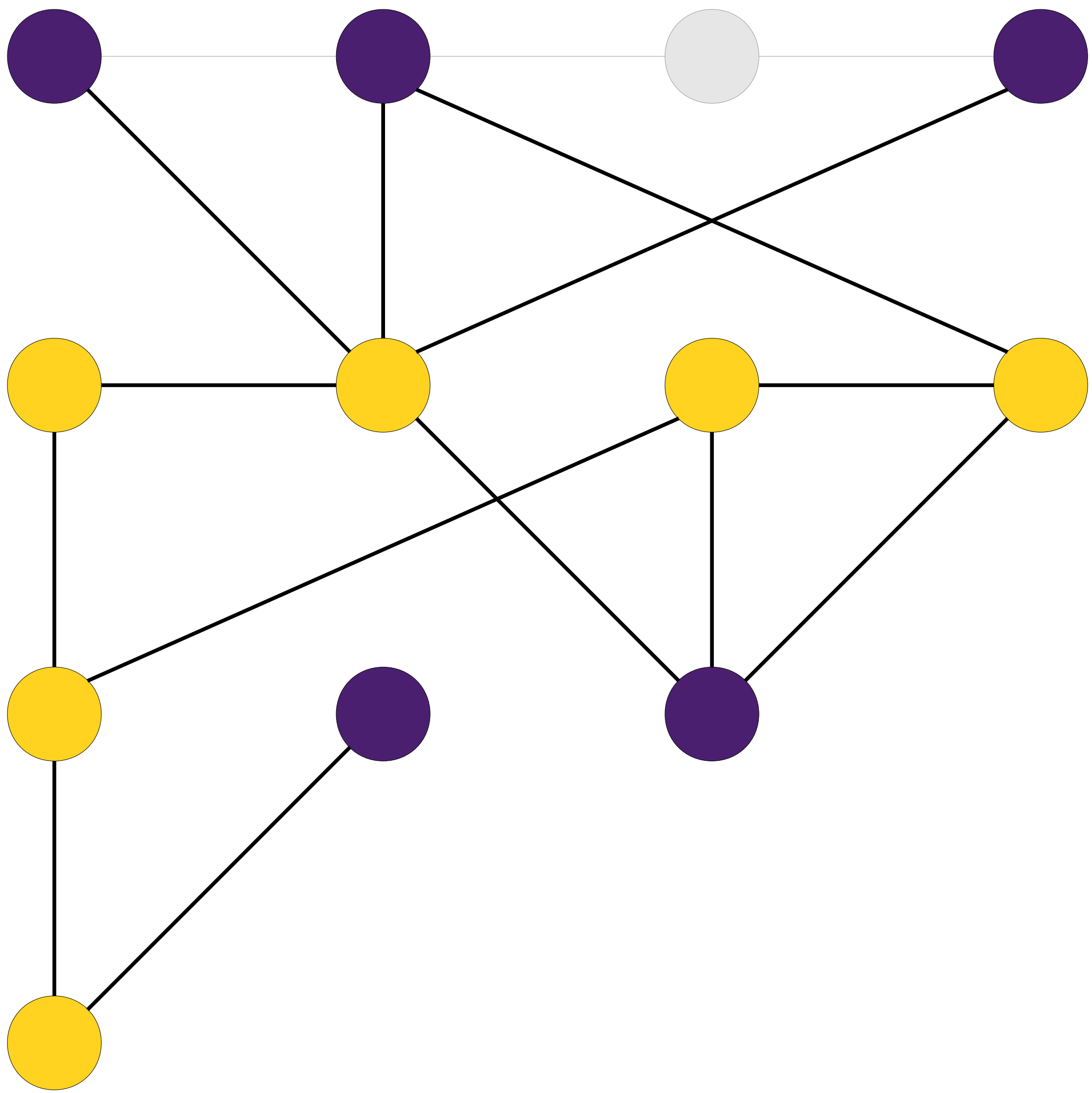
Iterating over a Graph



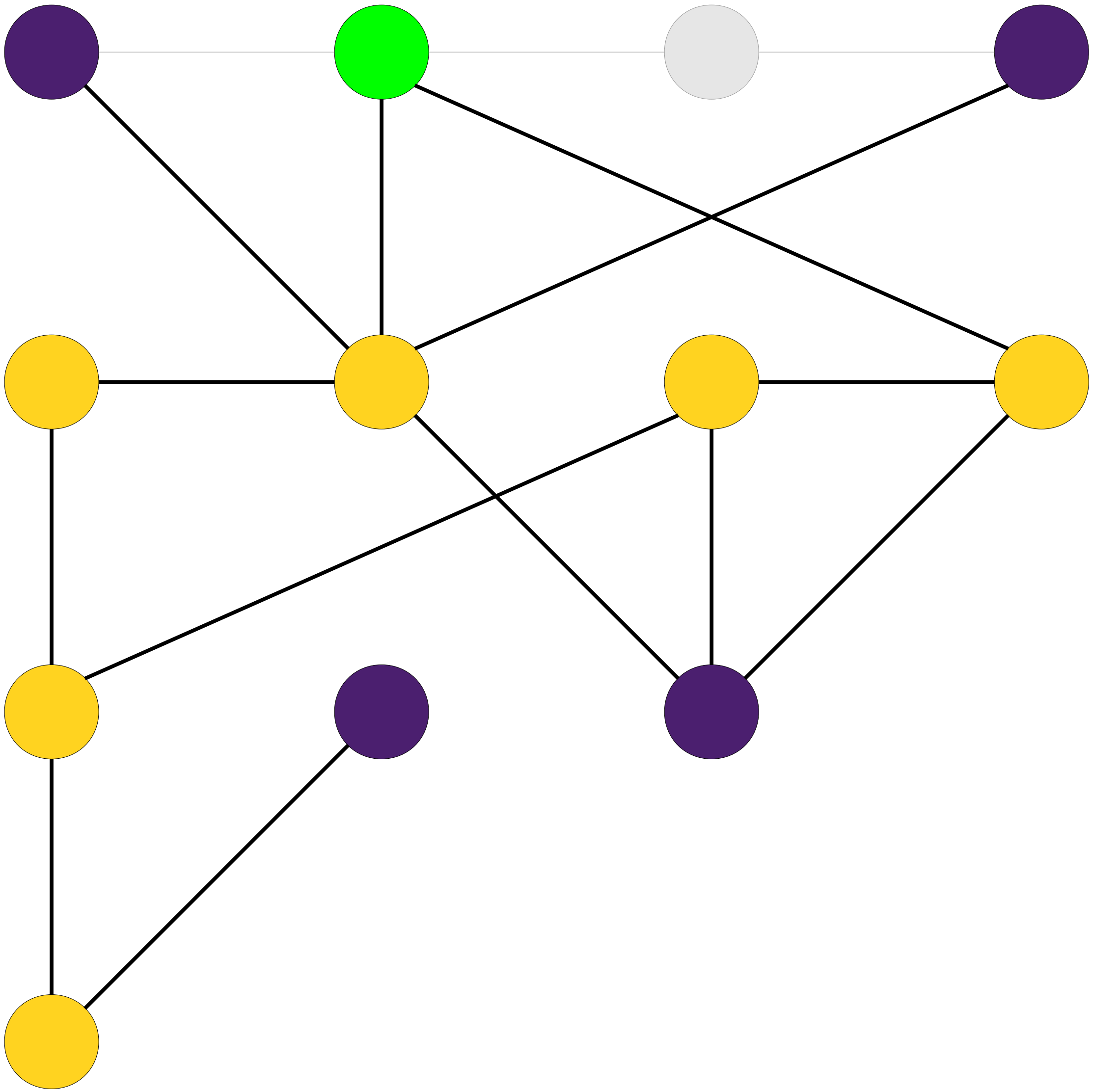
Iterating over a Graph



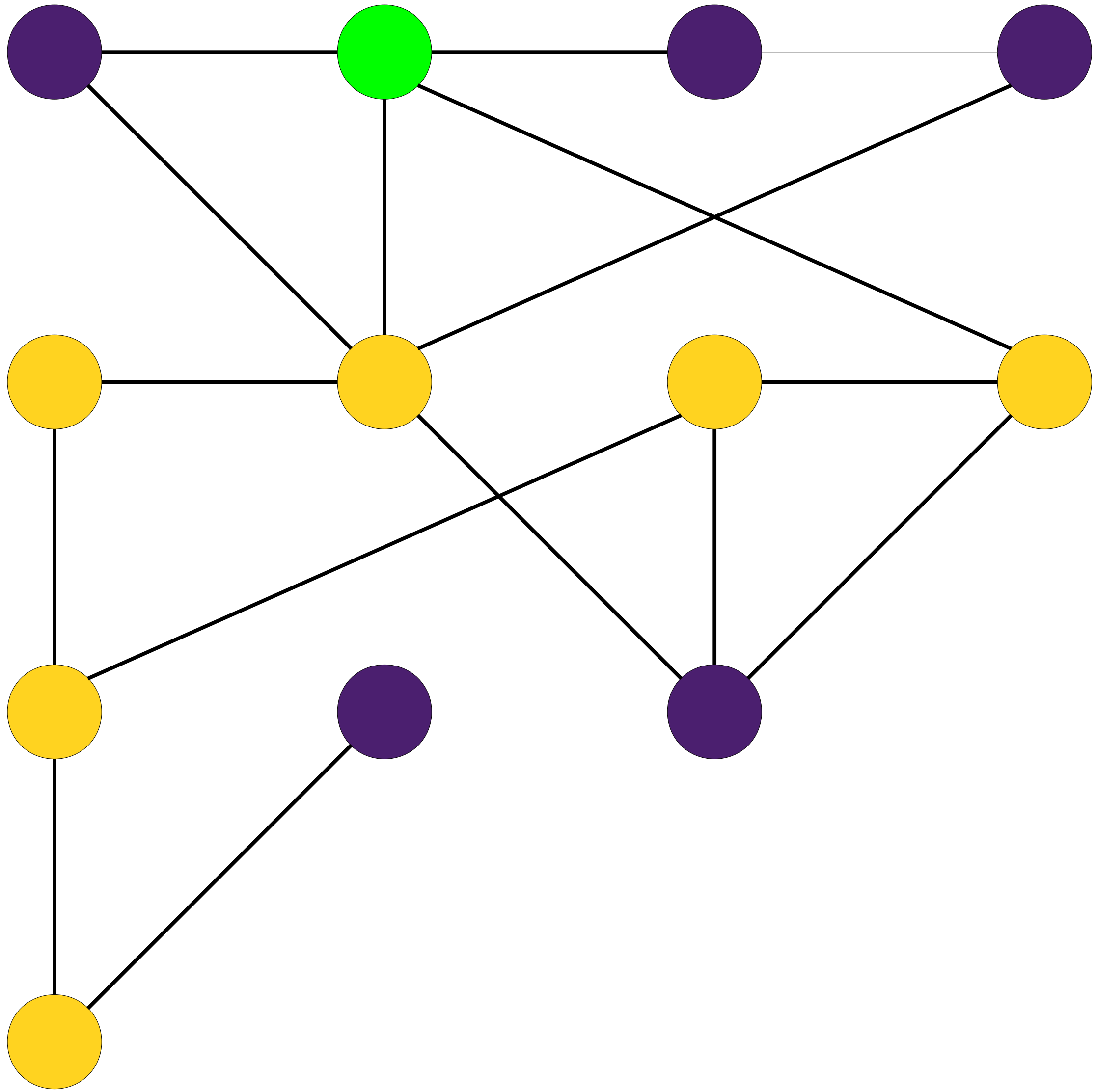
Iterating over a Graph



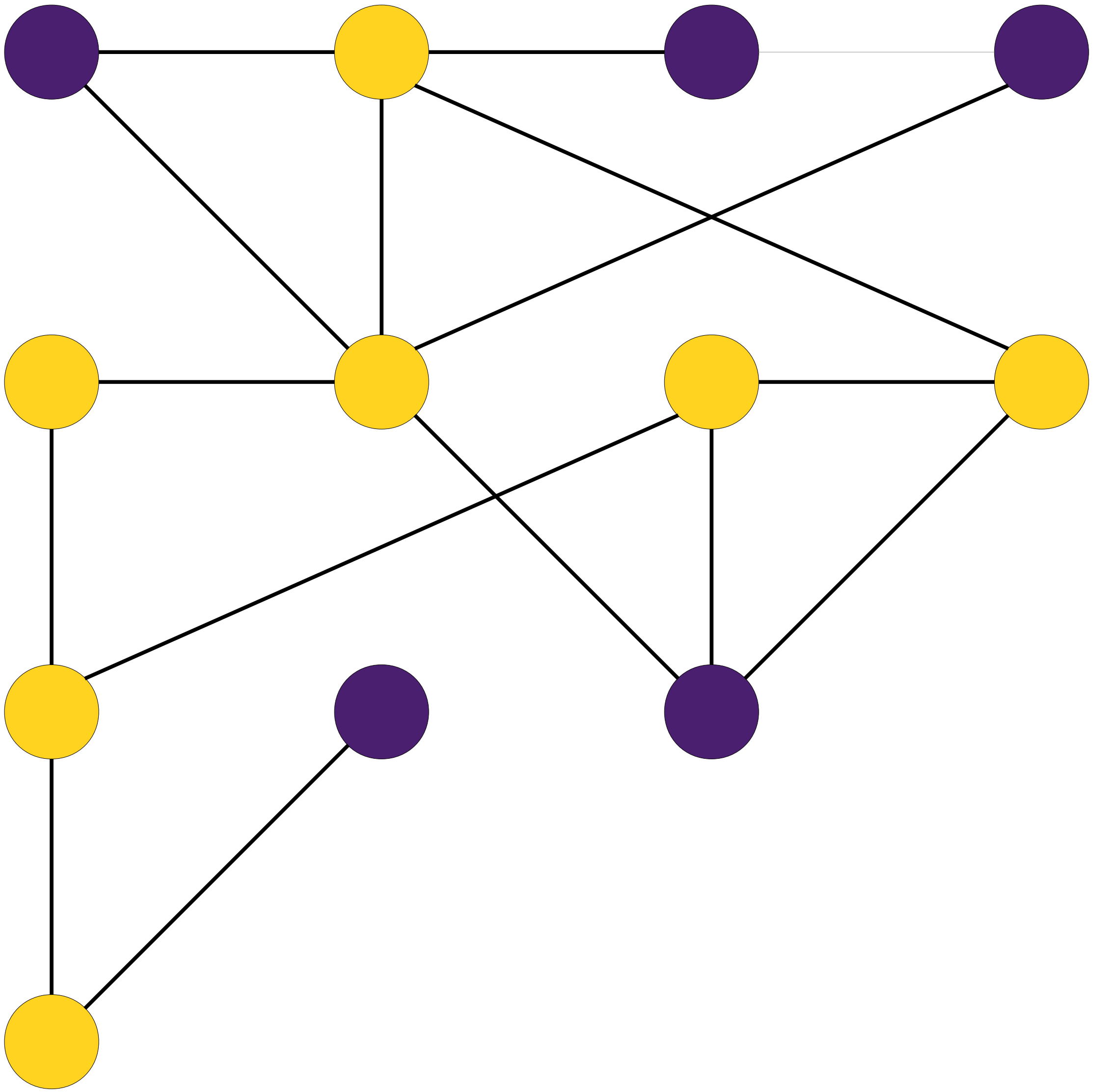
Iterating over a Graph



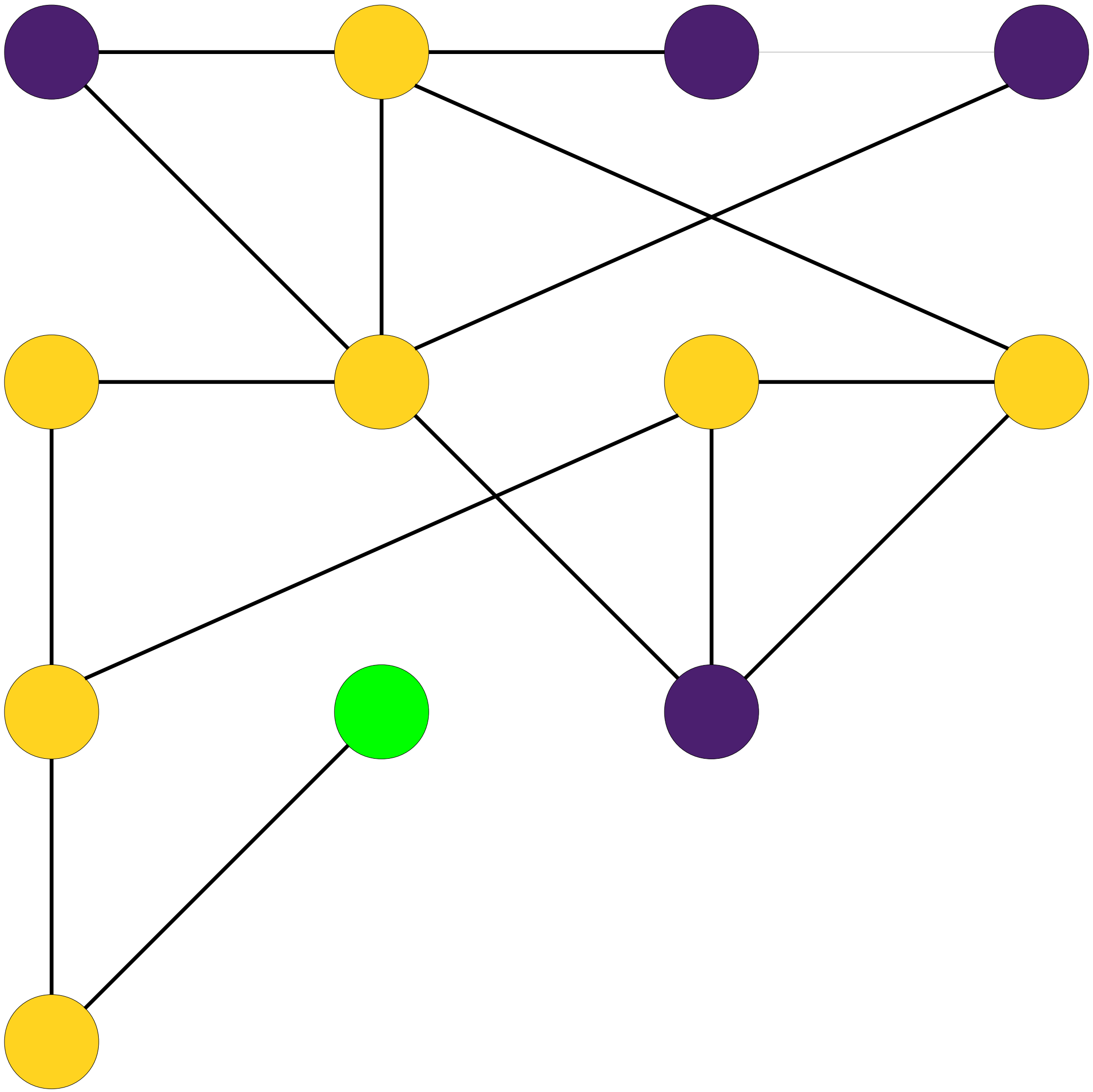
Iterating over a Graph



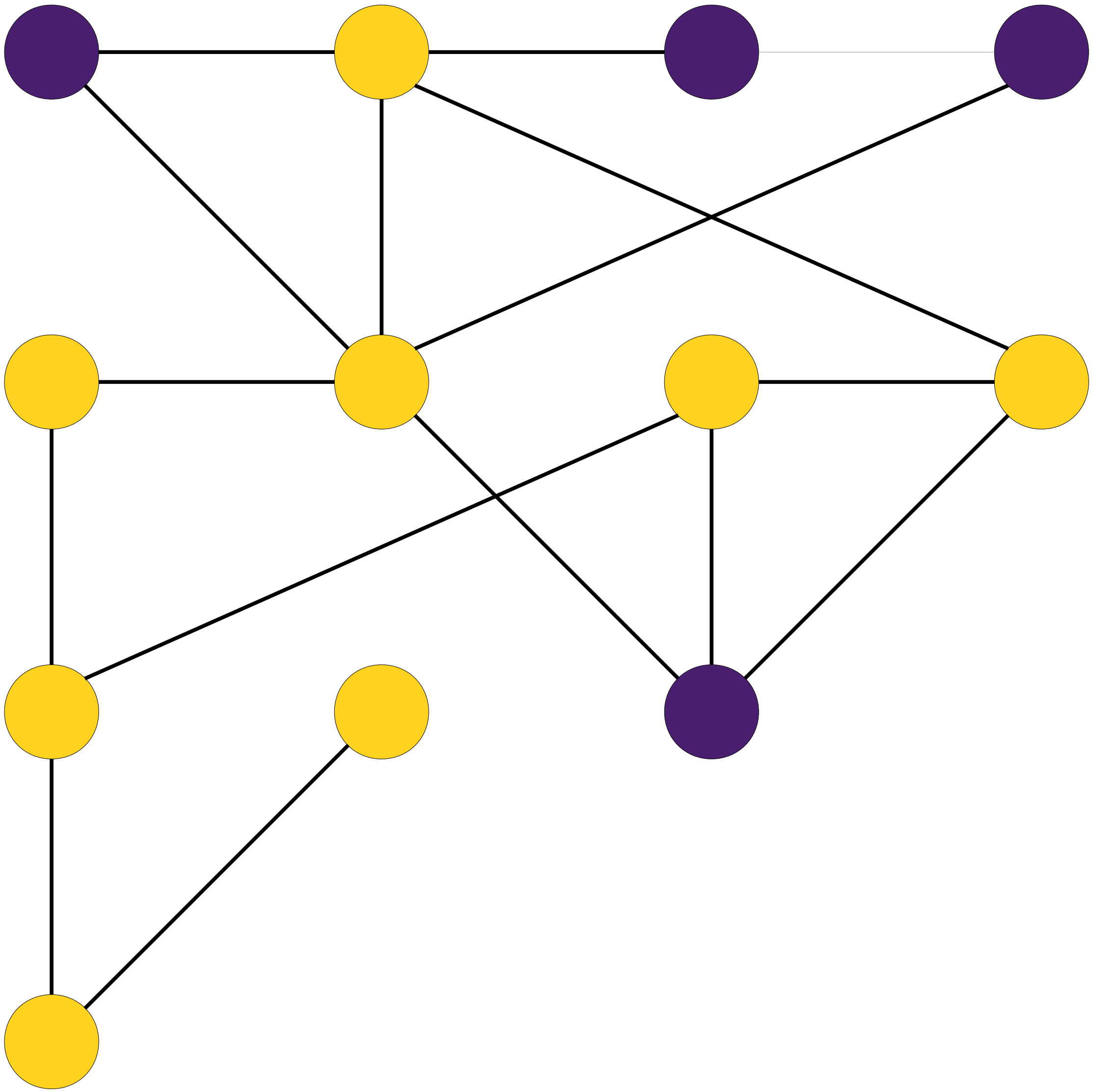
Iterating over a Graph



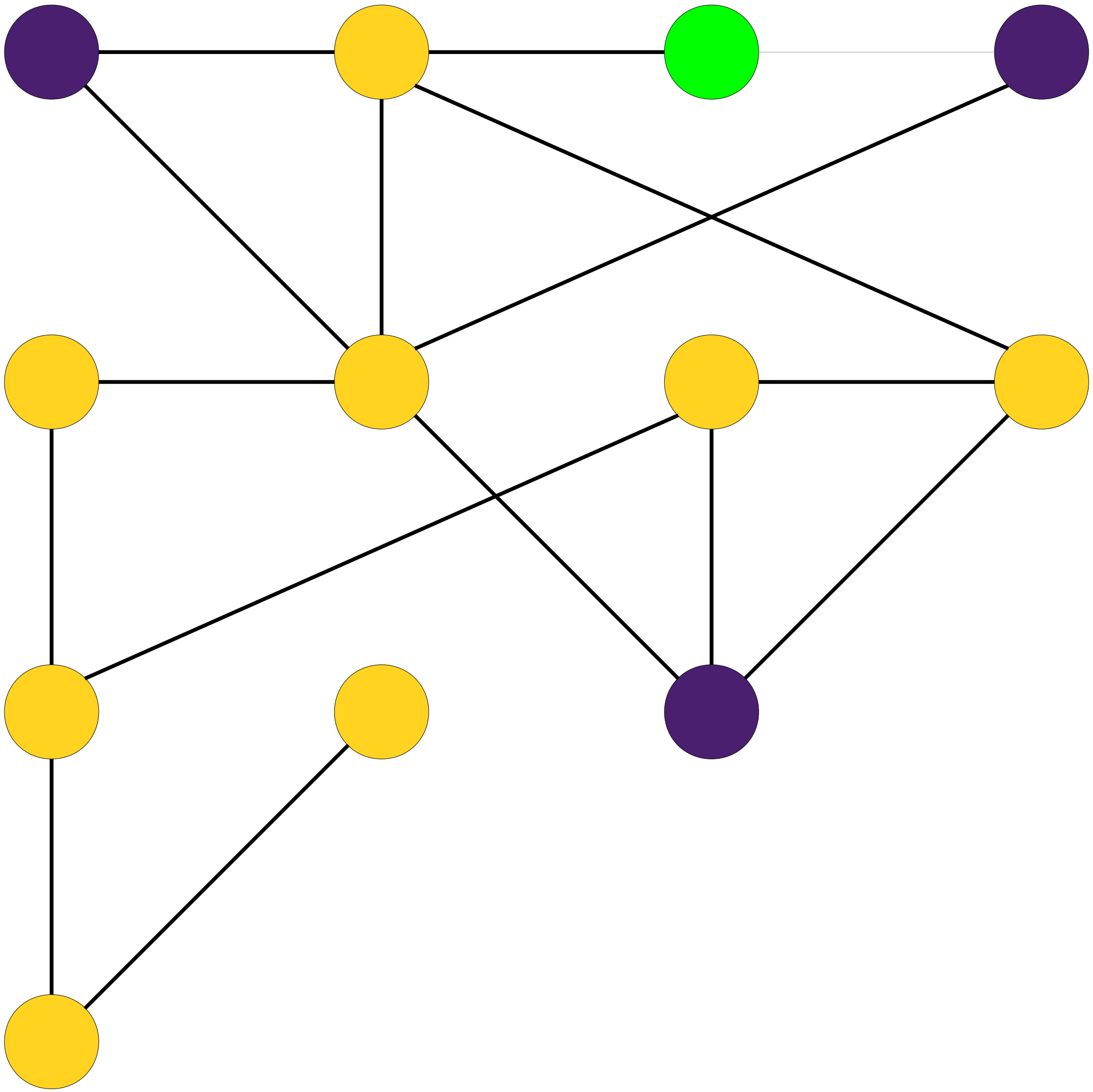
Iterating over a Graph



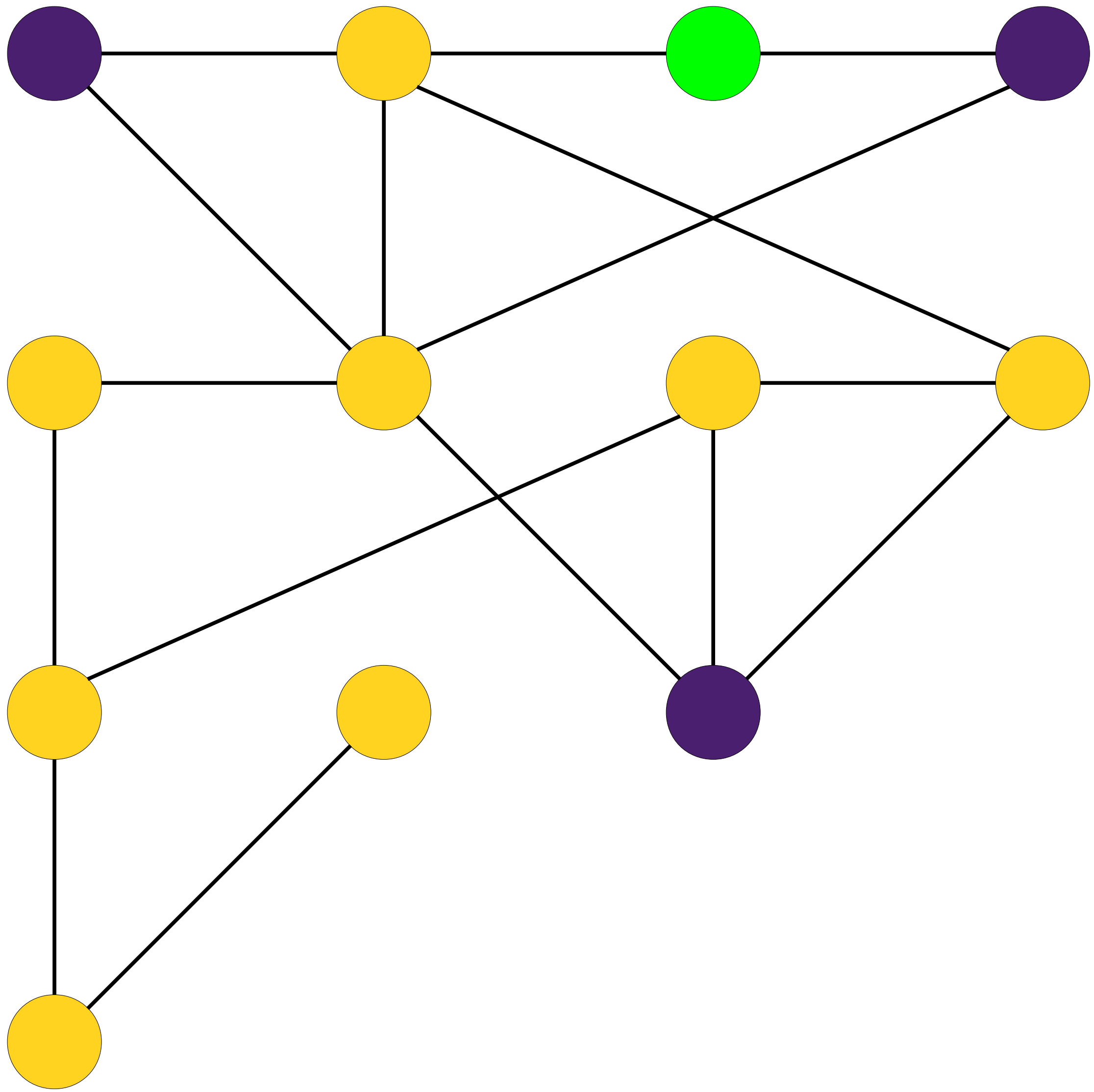
Iterating over a Graph



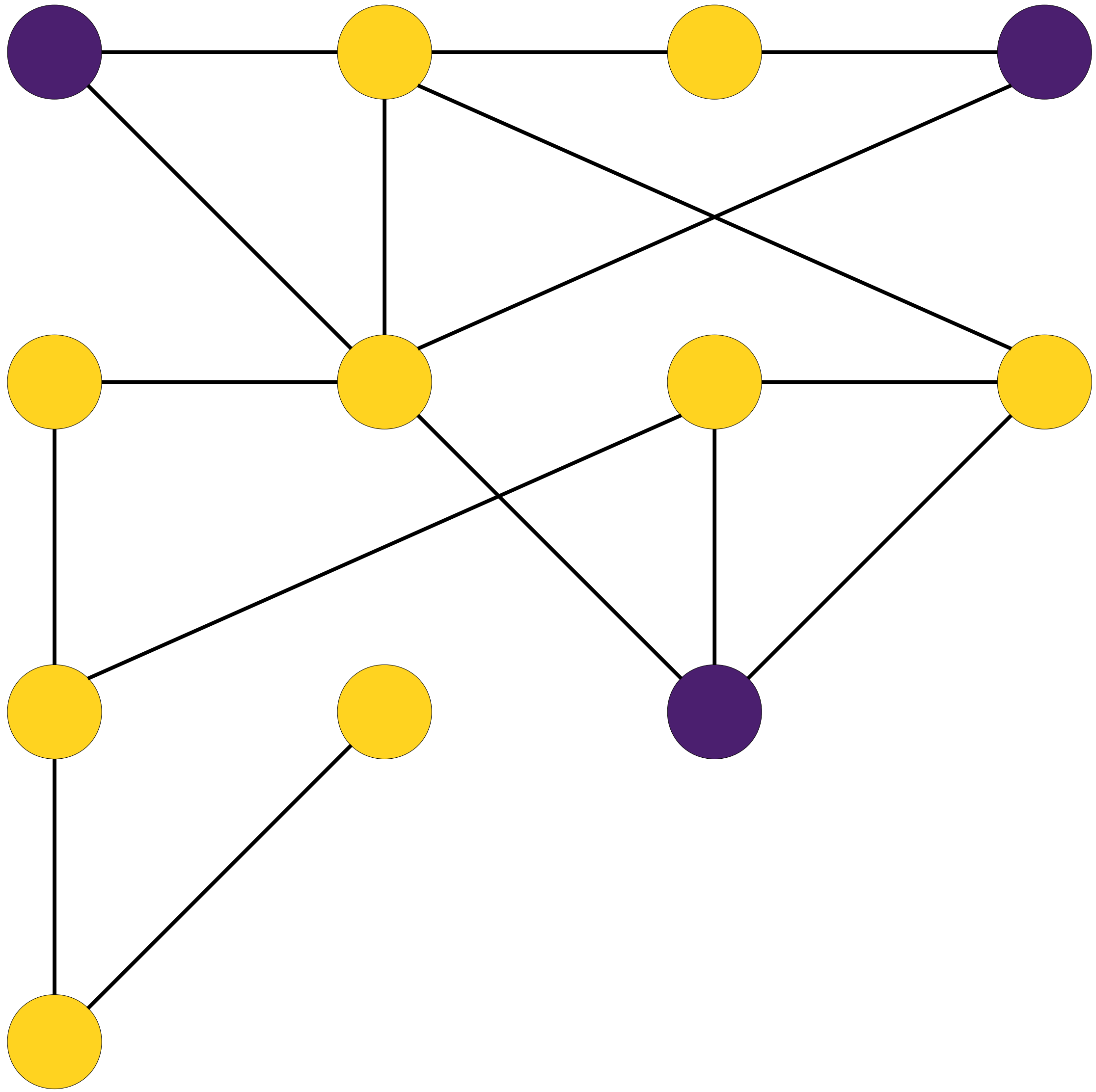
Iterating over a Graph



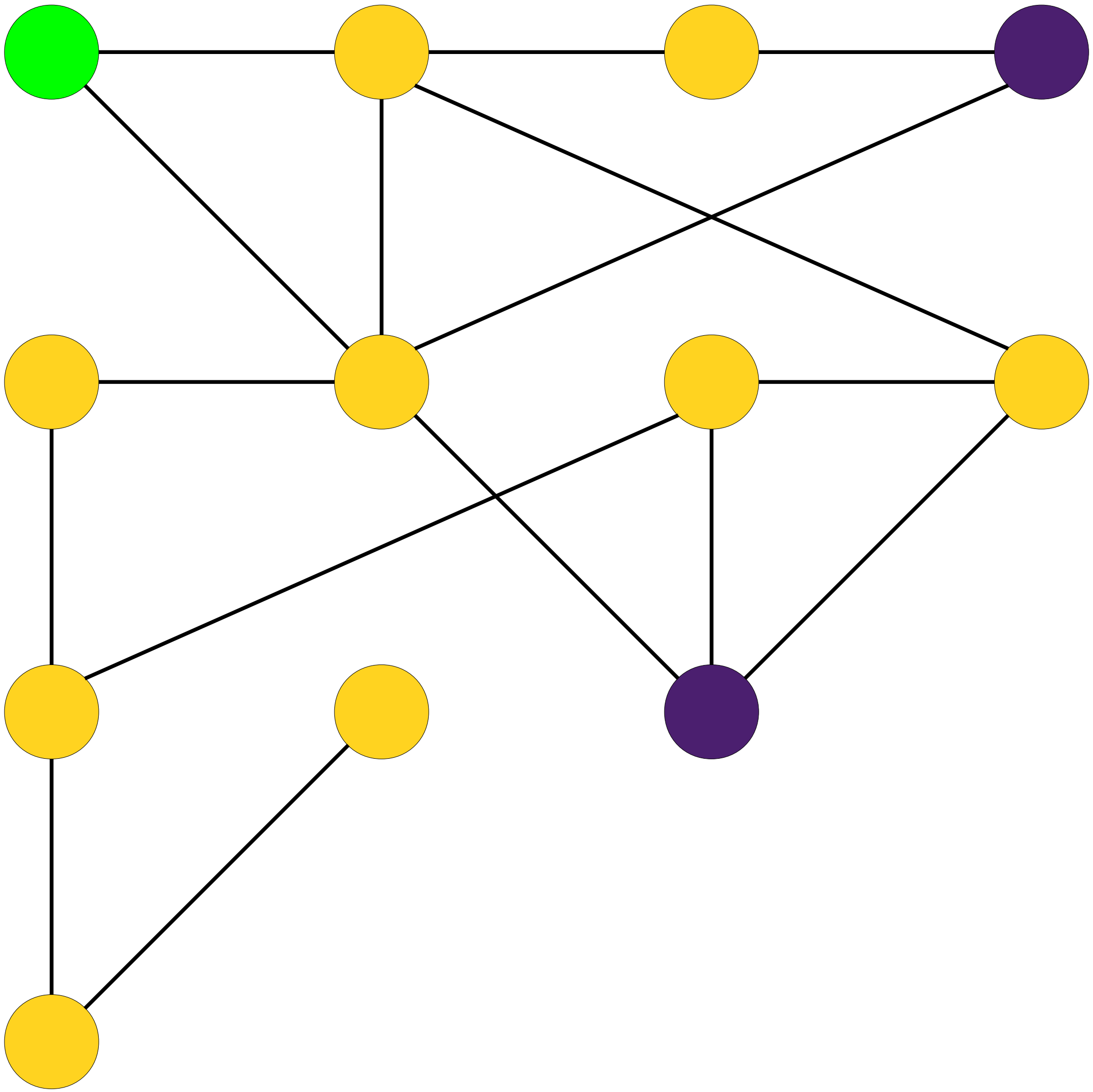
Iterating over a Graph



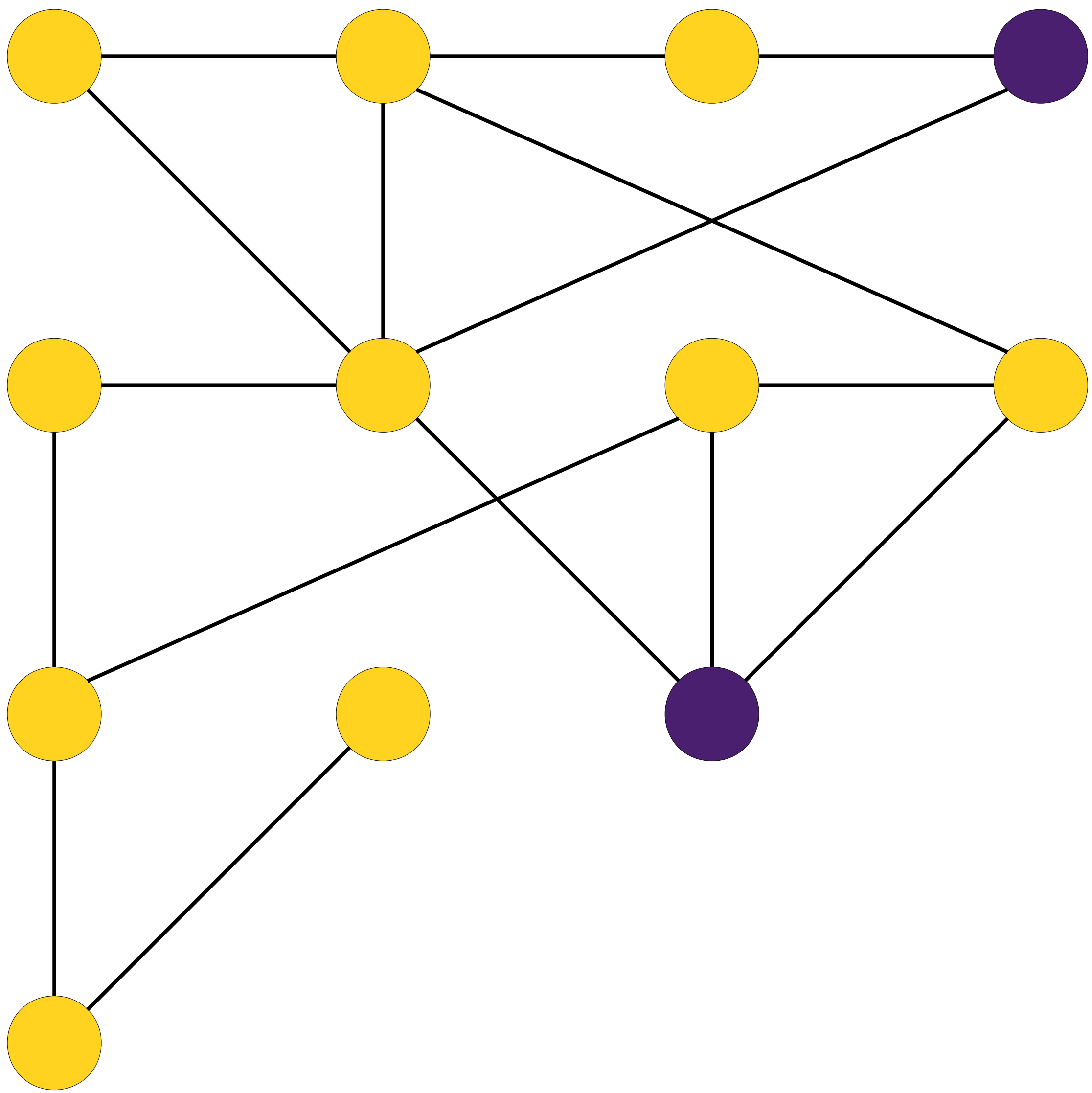
Iterating over a Graph



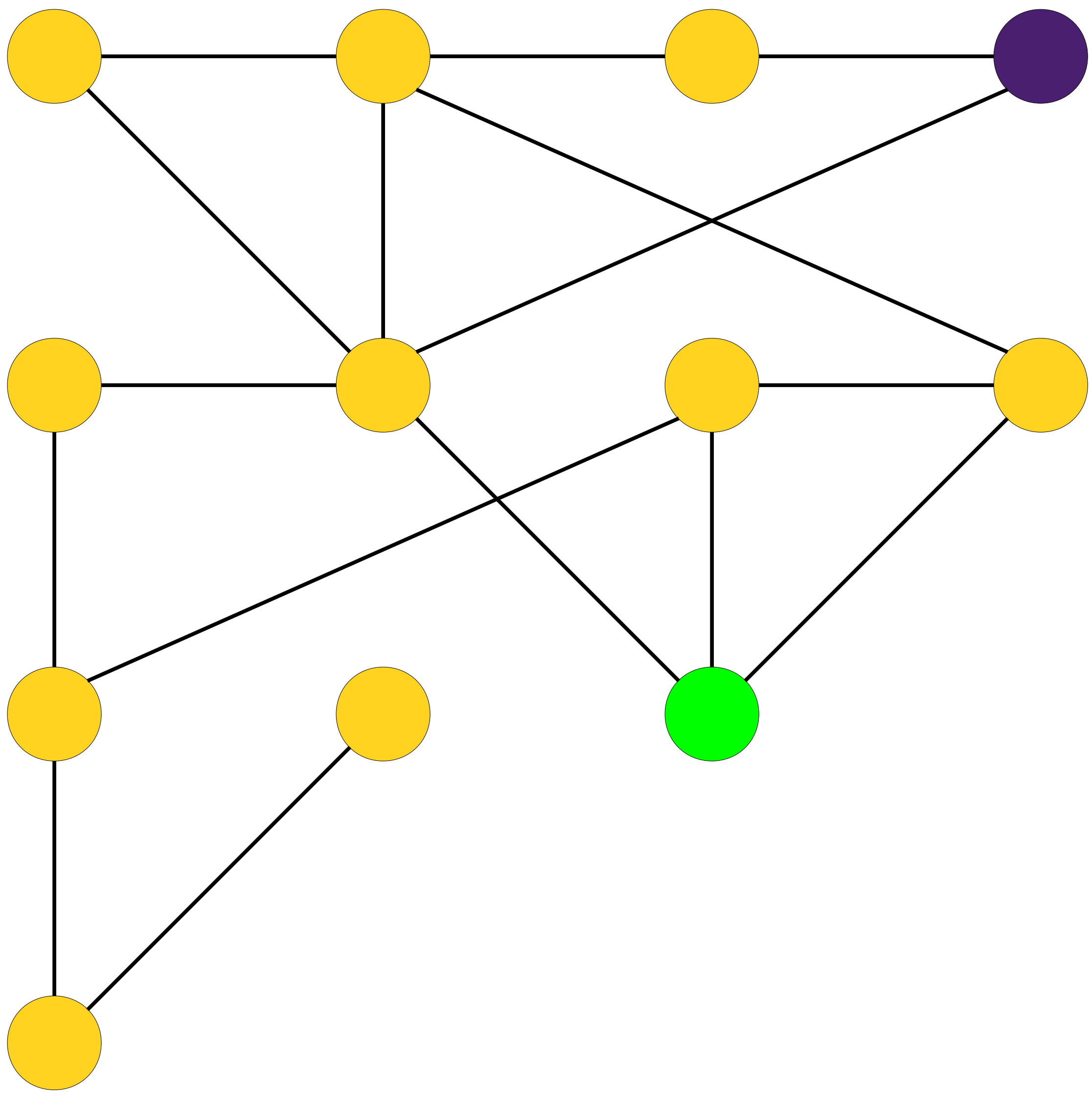
Iterating over a Graph



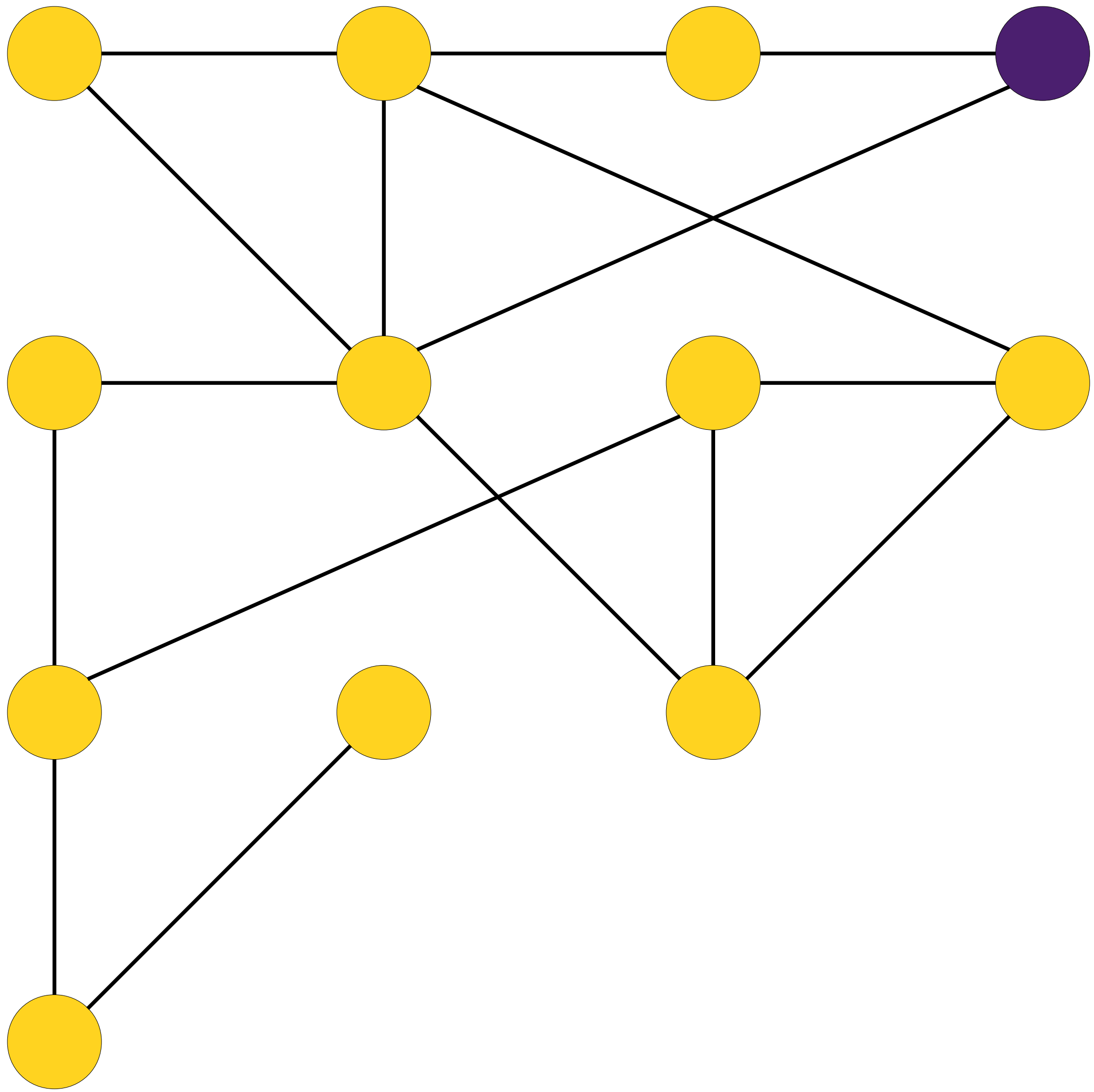
Iterating over a Graph



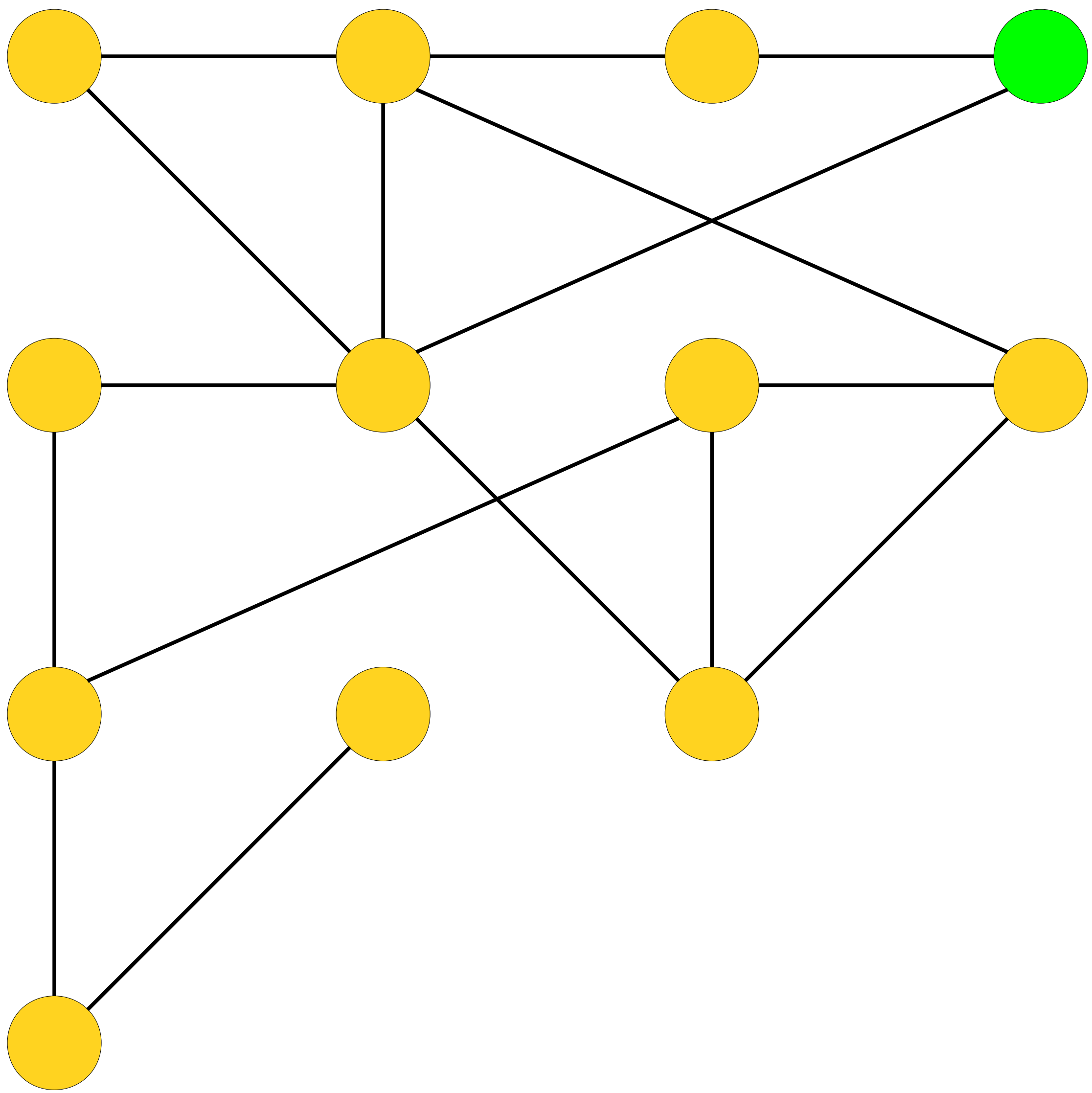
Iterating over a Graph



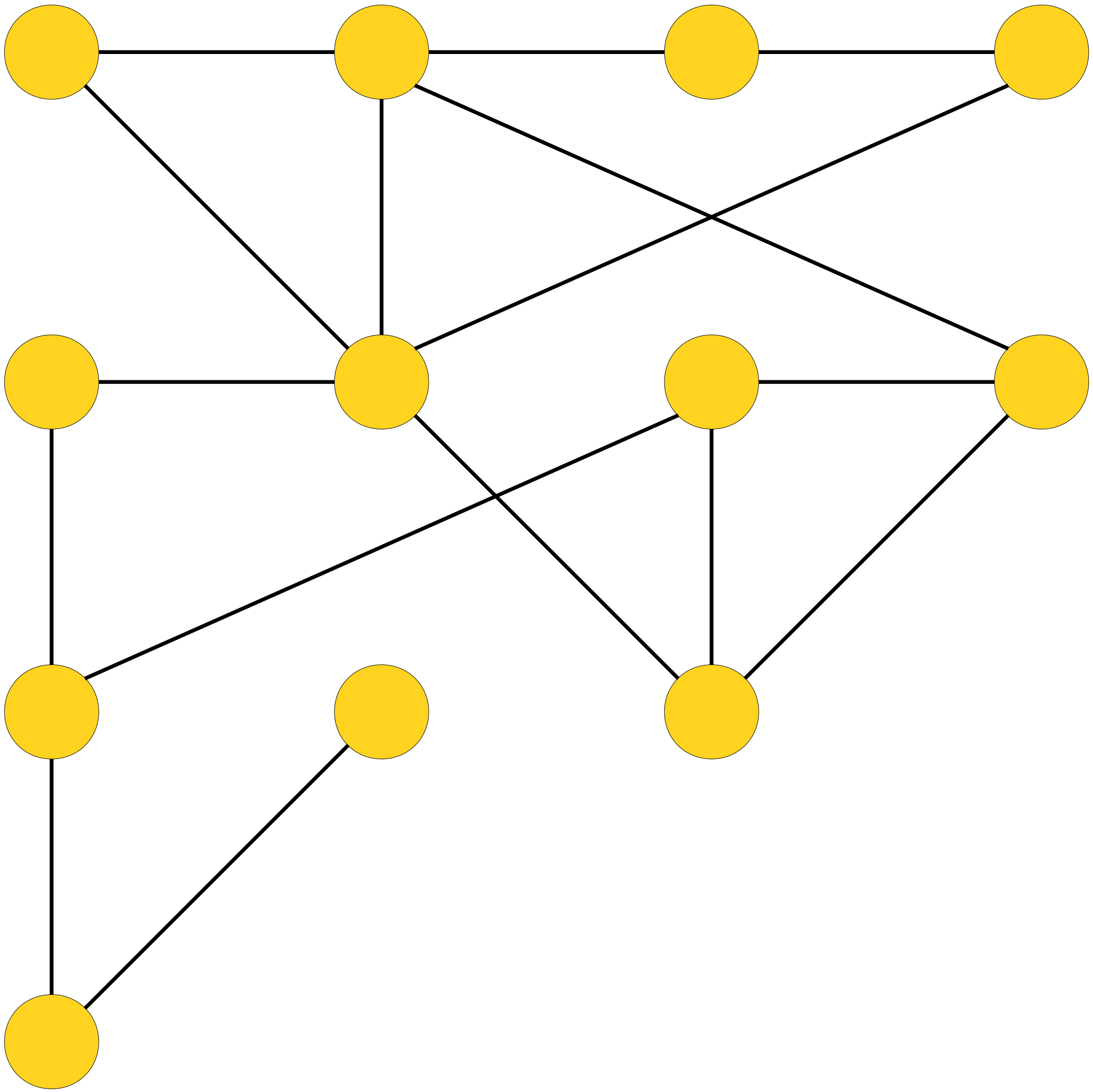
Iterating over a Graph



Iterating over a Graph



Iterating over a Graph



Iterating over a Graph

- All methods of iterating over a graph involve keeping track of 3 sets of nodes:

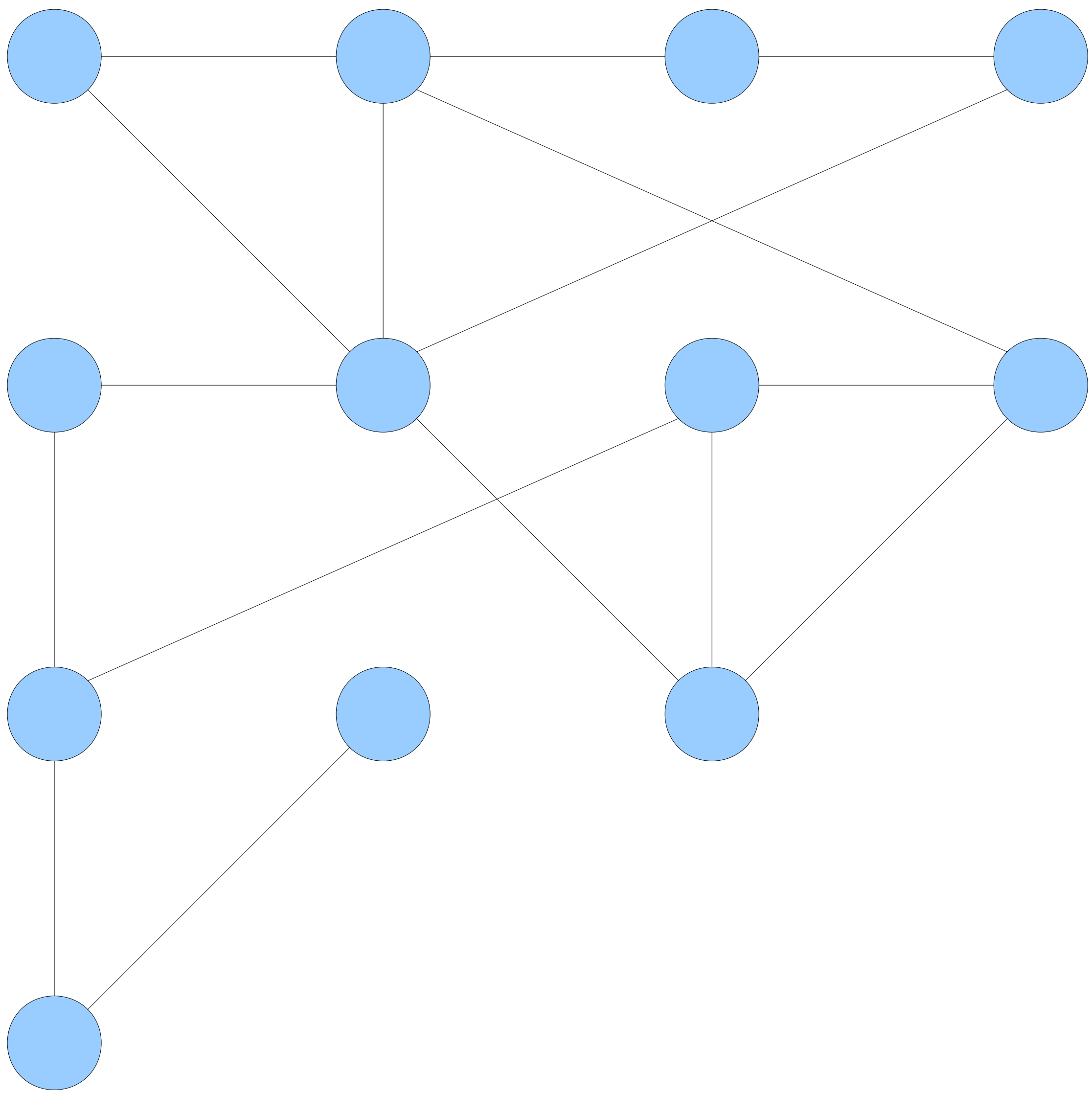
  - Set of Nodes already visited
  - Set of Nodes to look at next
  - Everything else

- Methods of iterating over nodes differ in how they choose which node to look at next

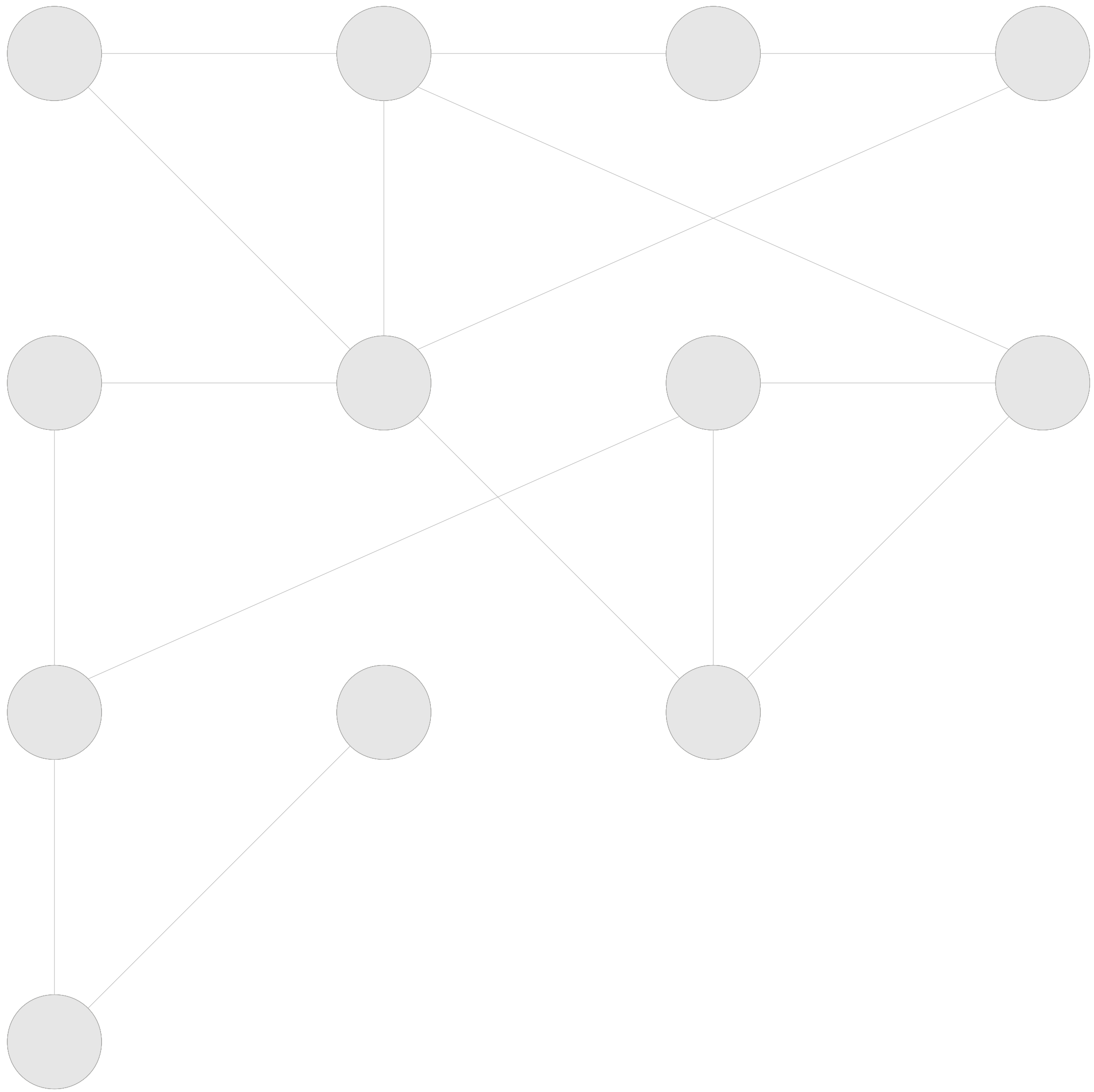
Recursive Depth First Search

- To detect connected components we just want to see whether or not some path exists between them (we don't care about finding the “shortest” path).
- One way to detect connectivity would be to just pick an arbitrary direction and keep following it.
  - When you run out of room to go in one direction, just go back and look in a different direction

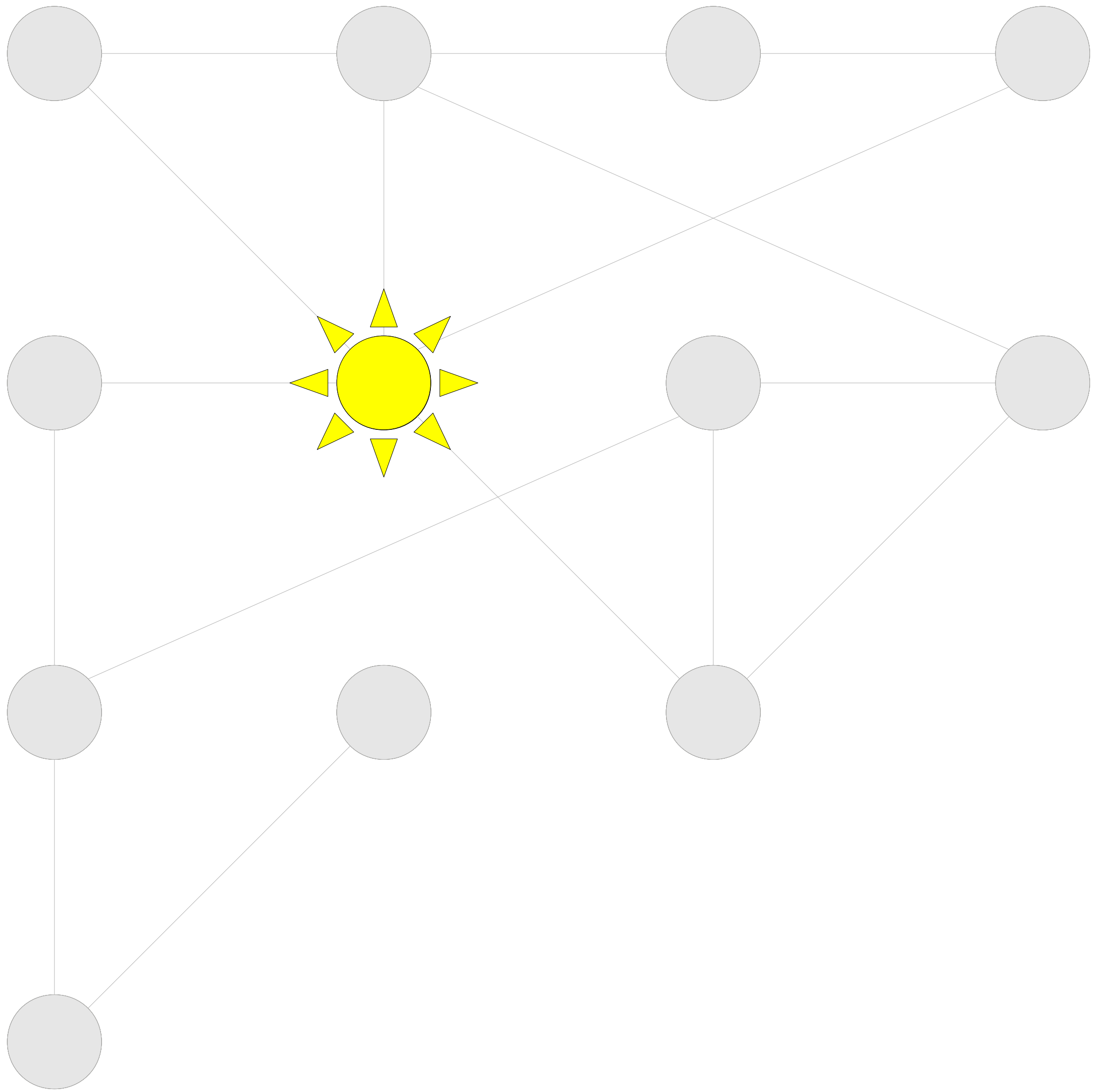
Recursive Depth-First Search



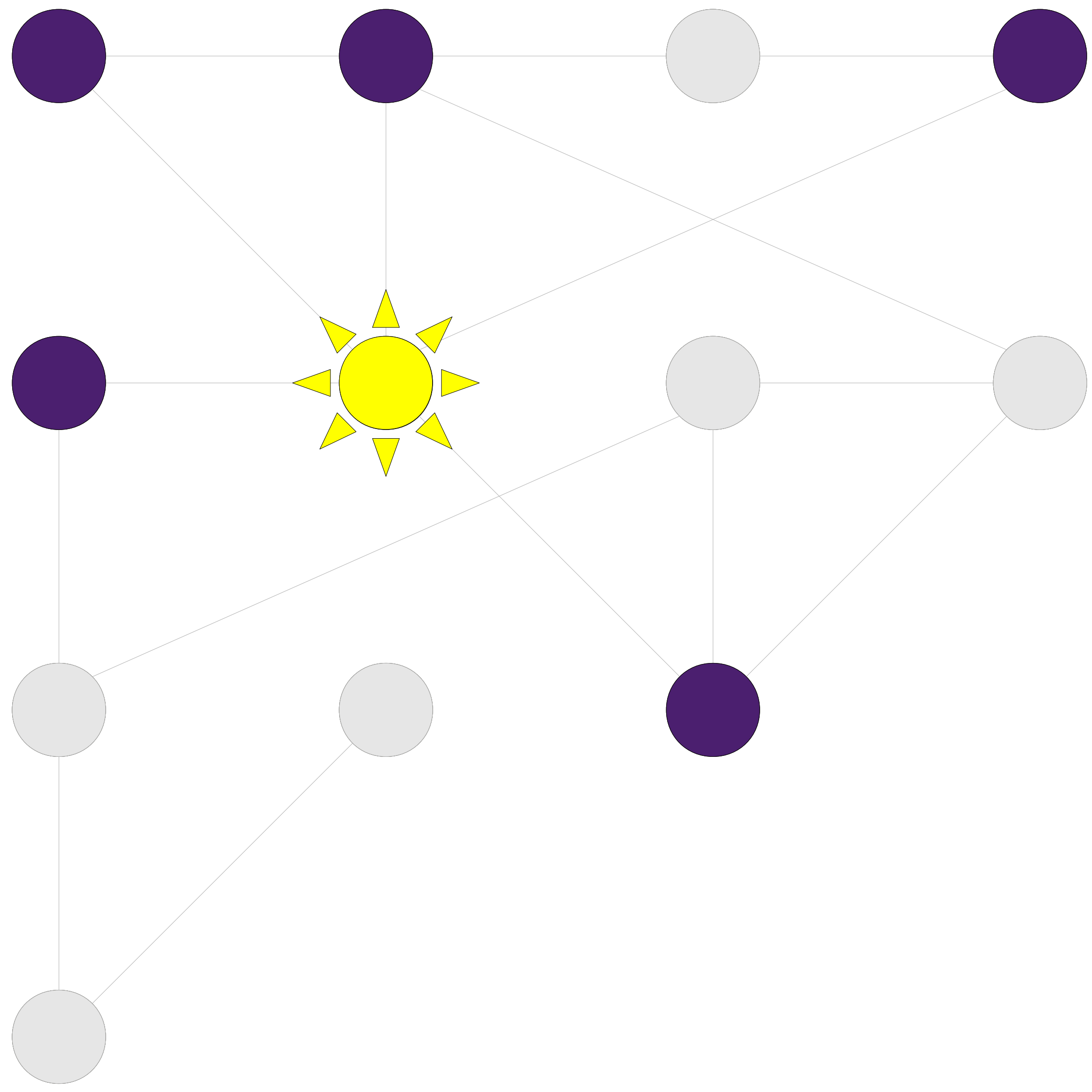
Recursive Depth-First Search



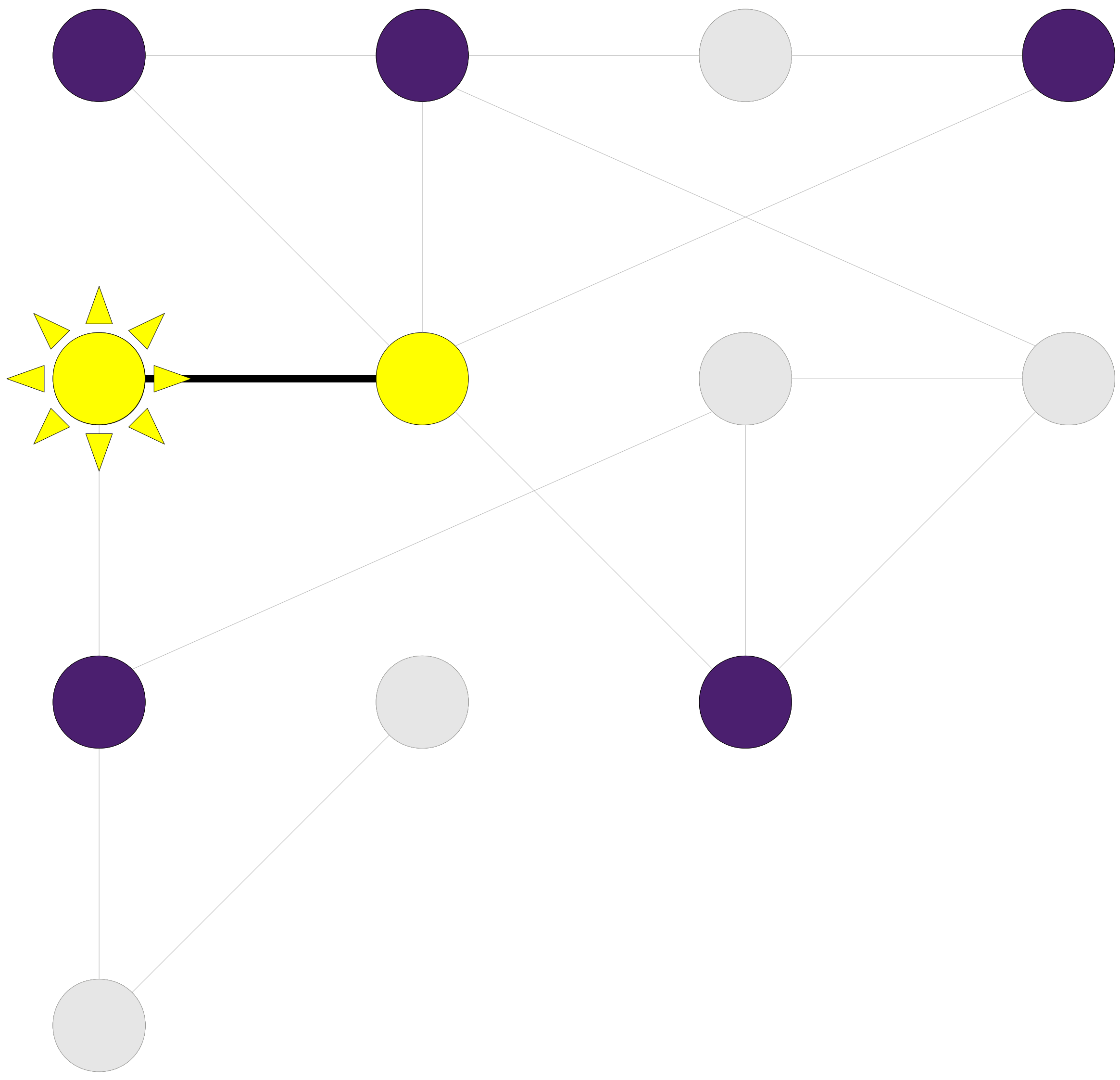
Recursive Depth-First Search



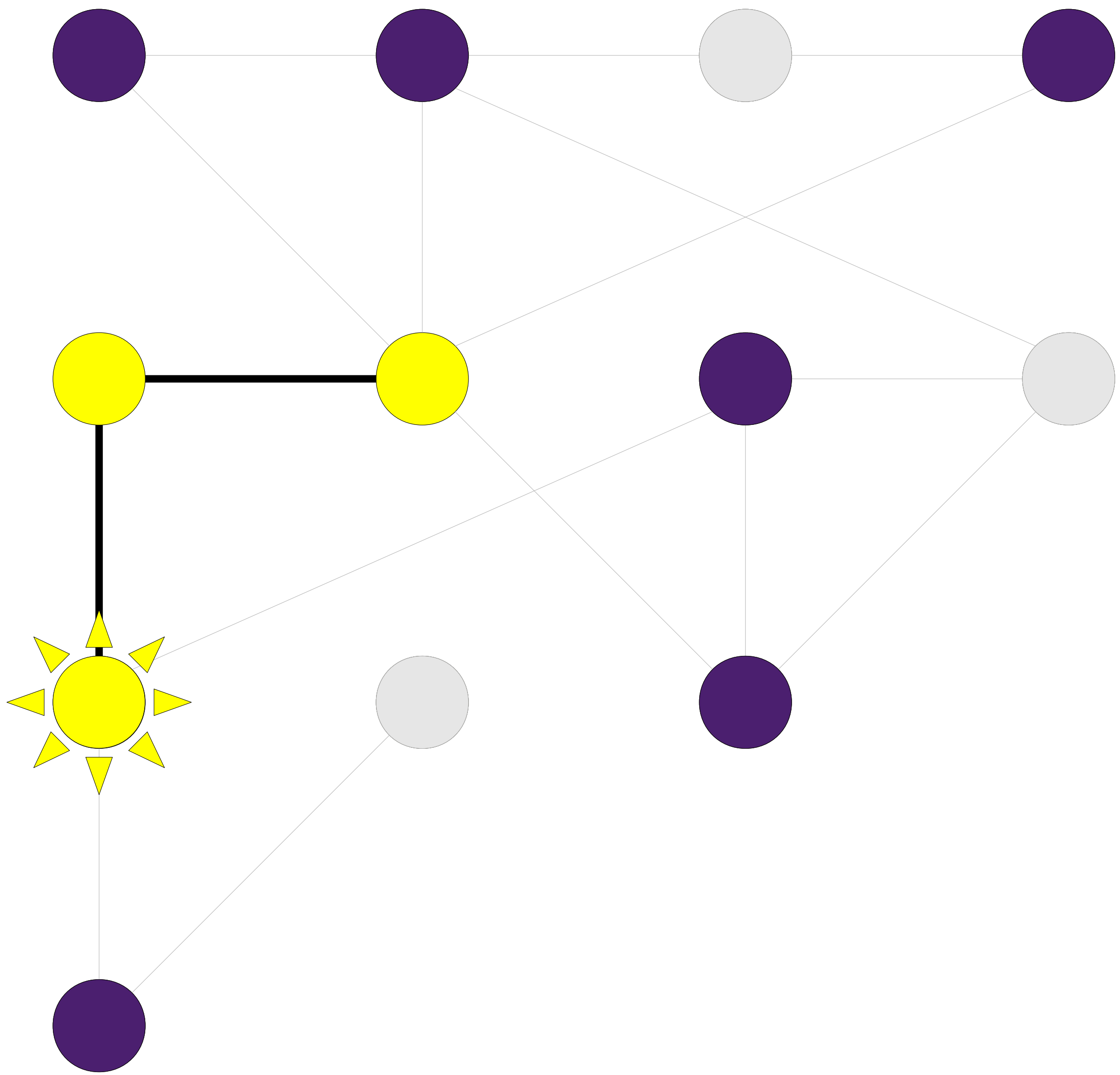
Recursive Depth-First Search



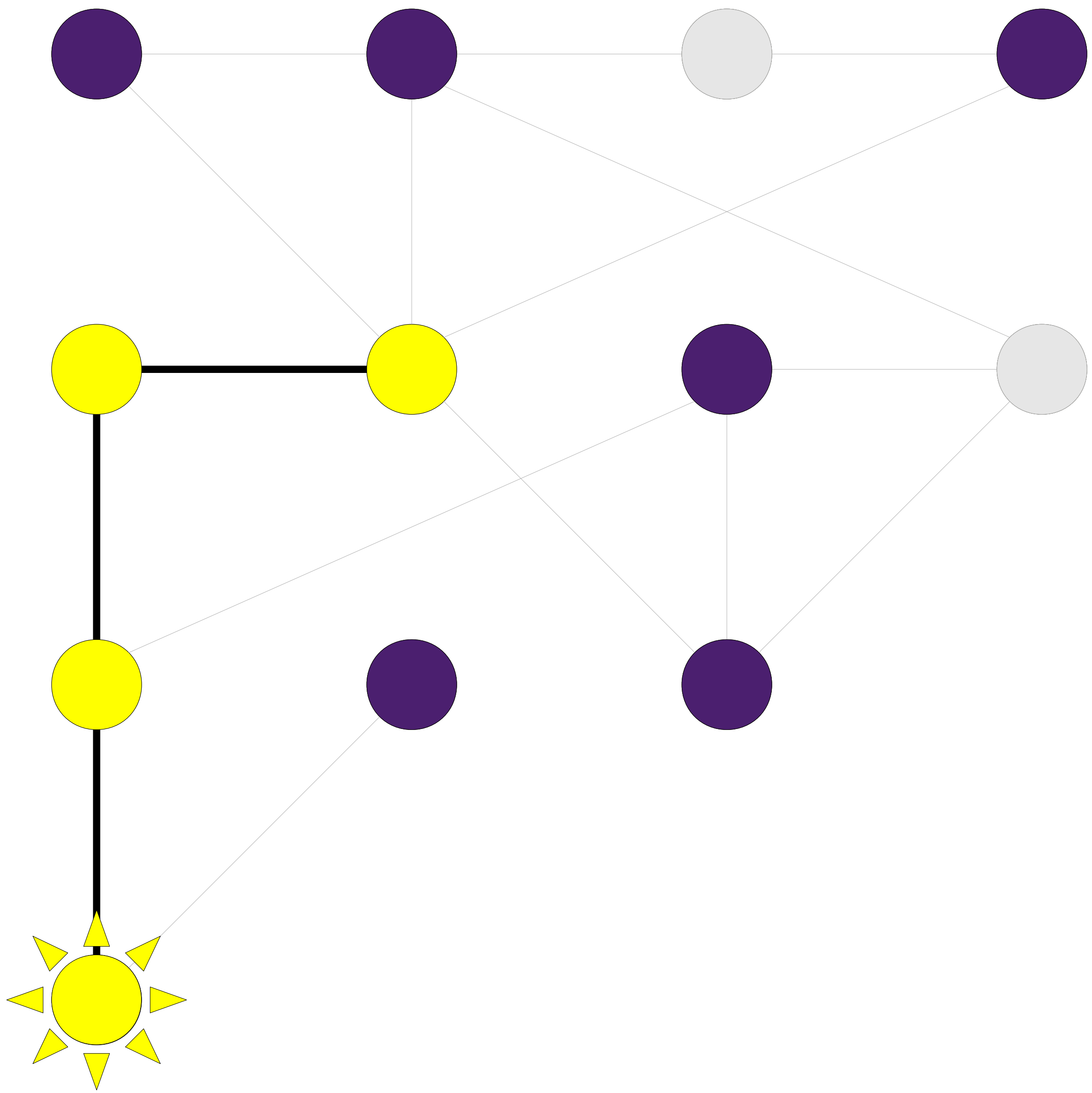
Recursive Depth-First Search



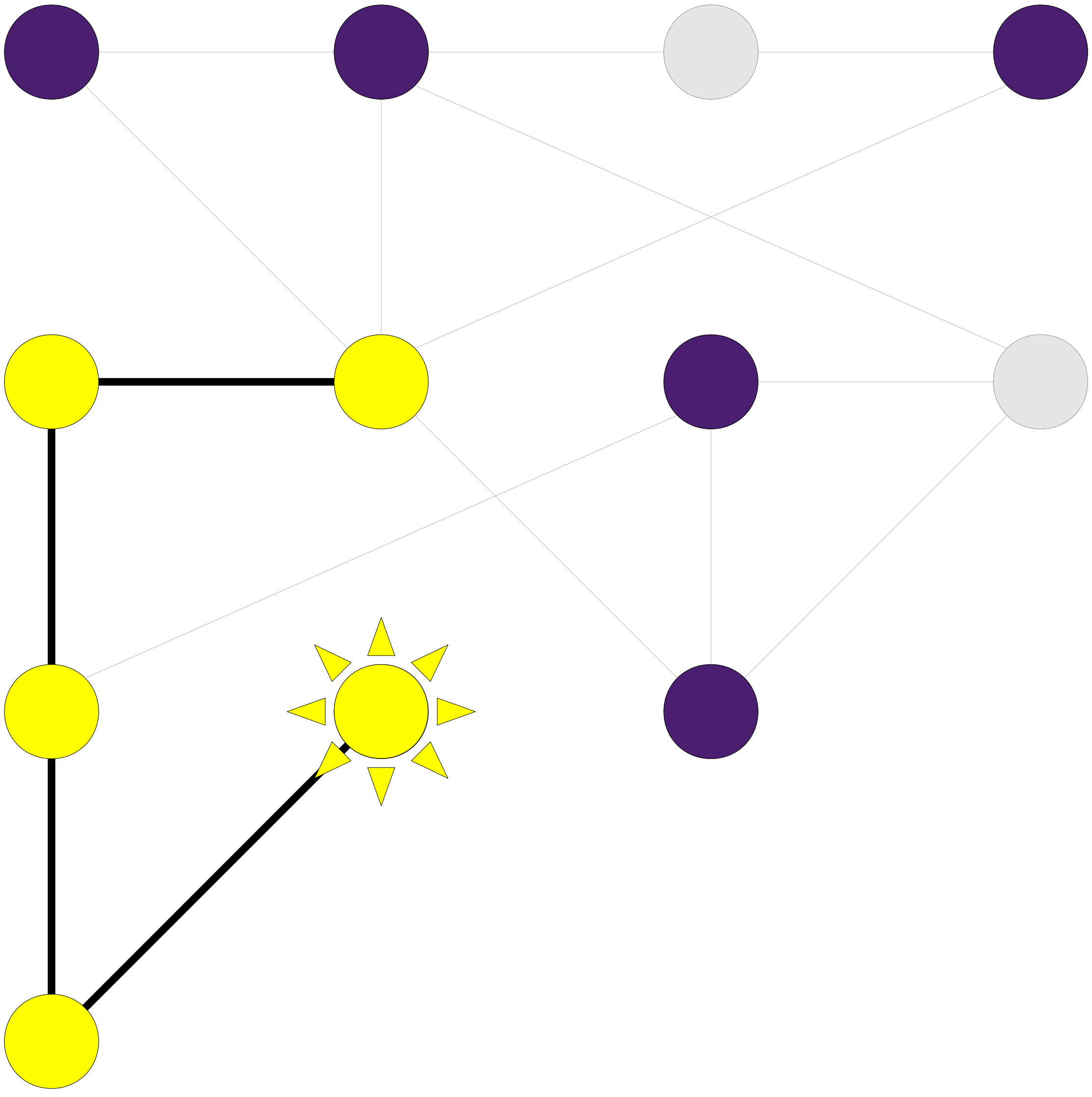
Recursive Depth-First Search



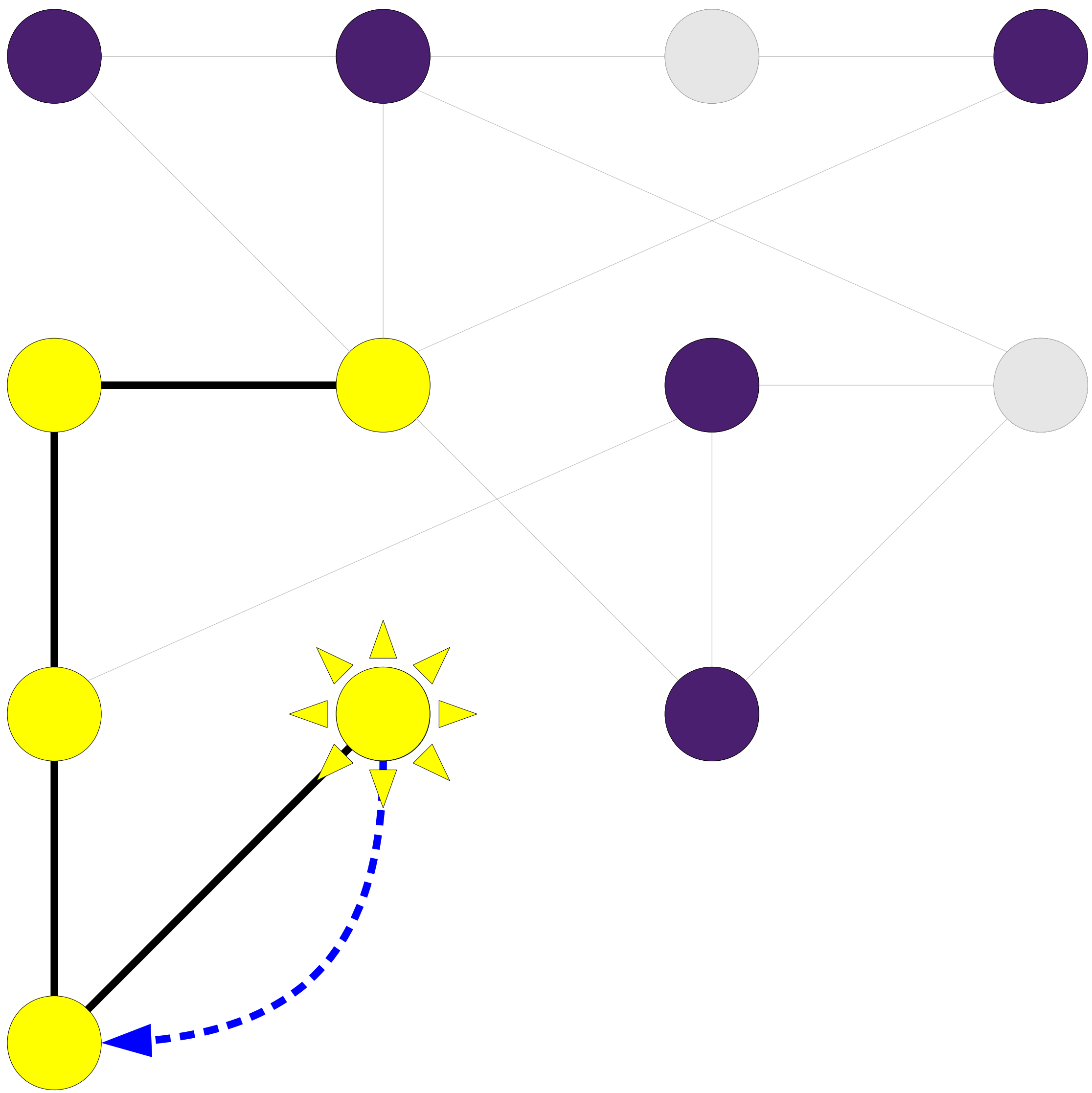
Recursive Depth-First Search



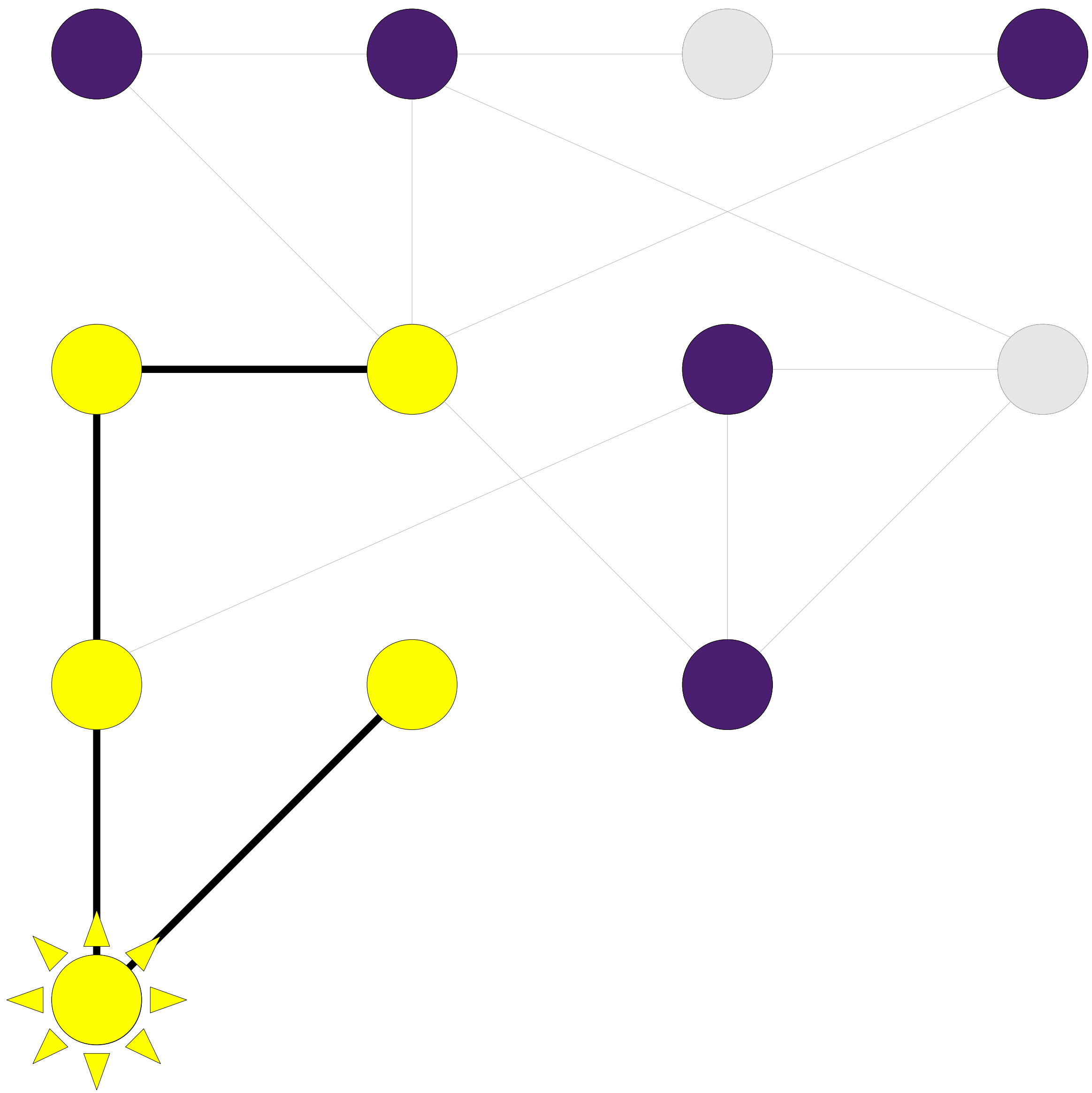
Recursive Depth-First Search



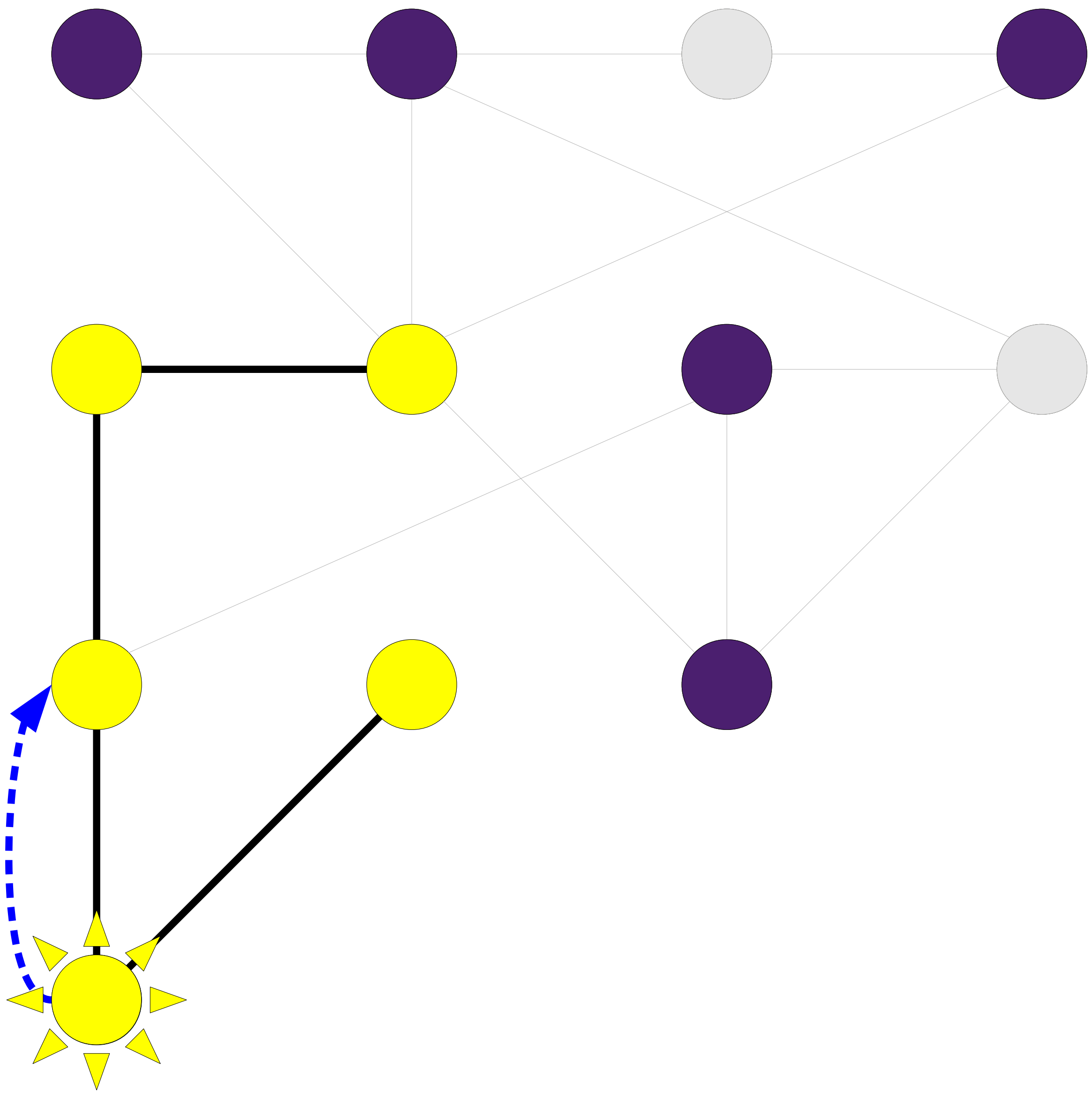
Recursive Depth-First Search



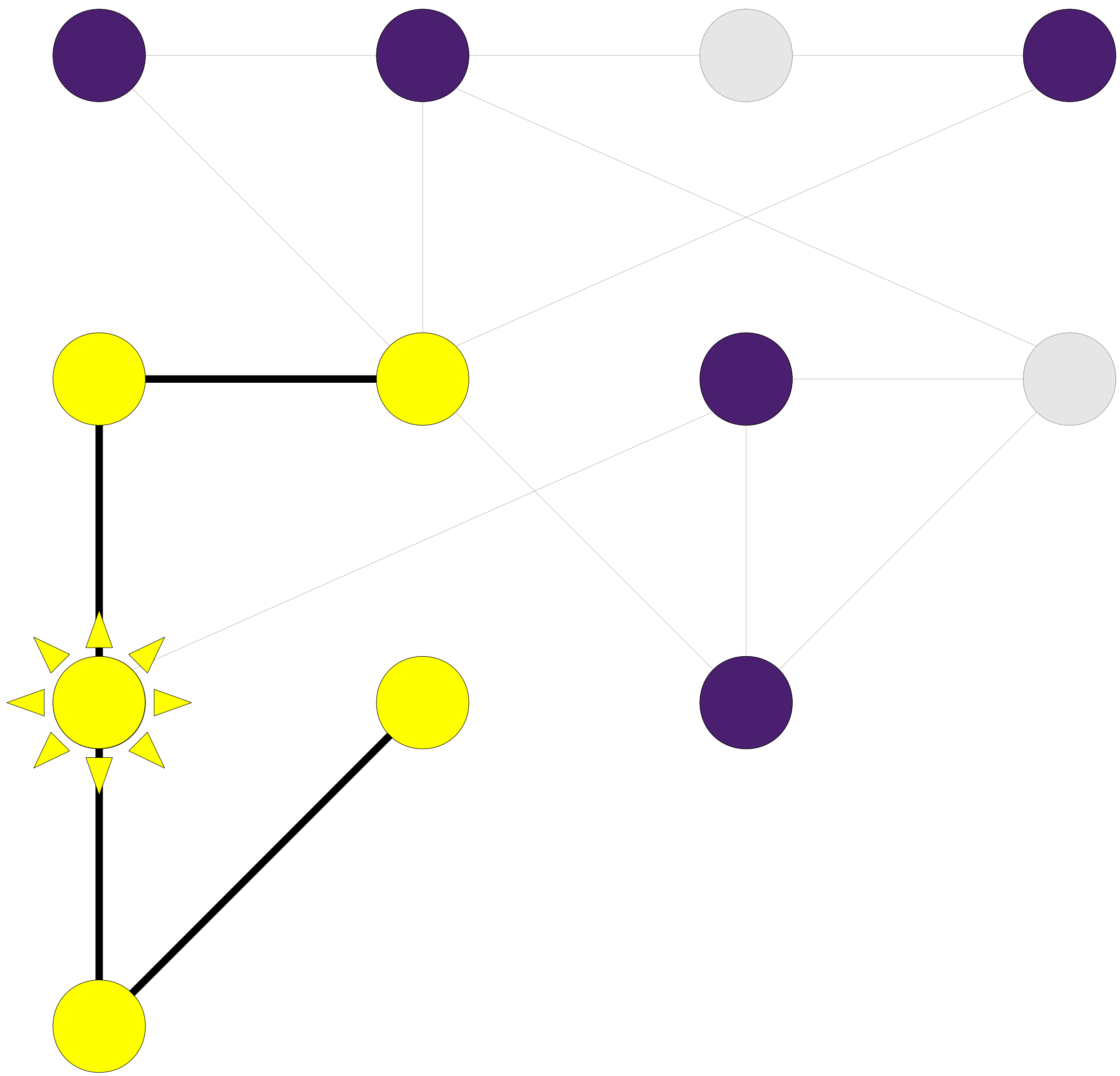
Recursive Depth-First Search



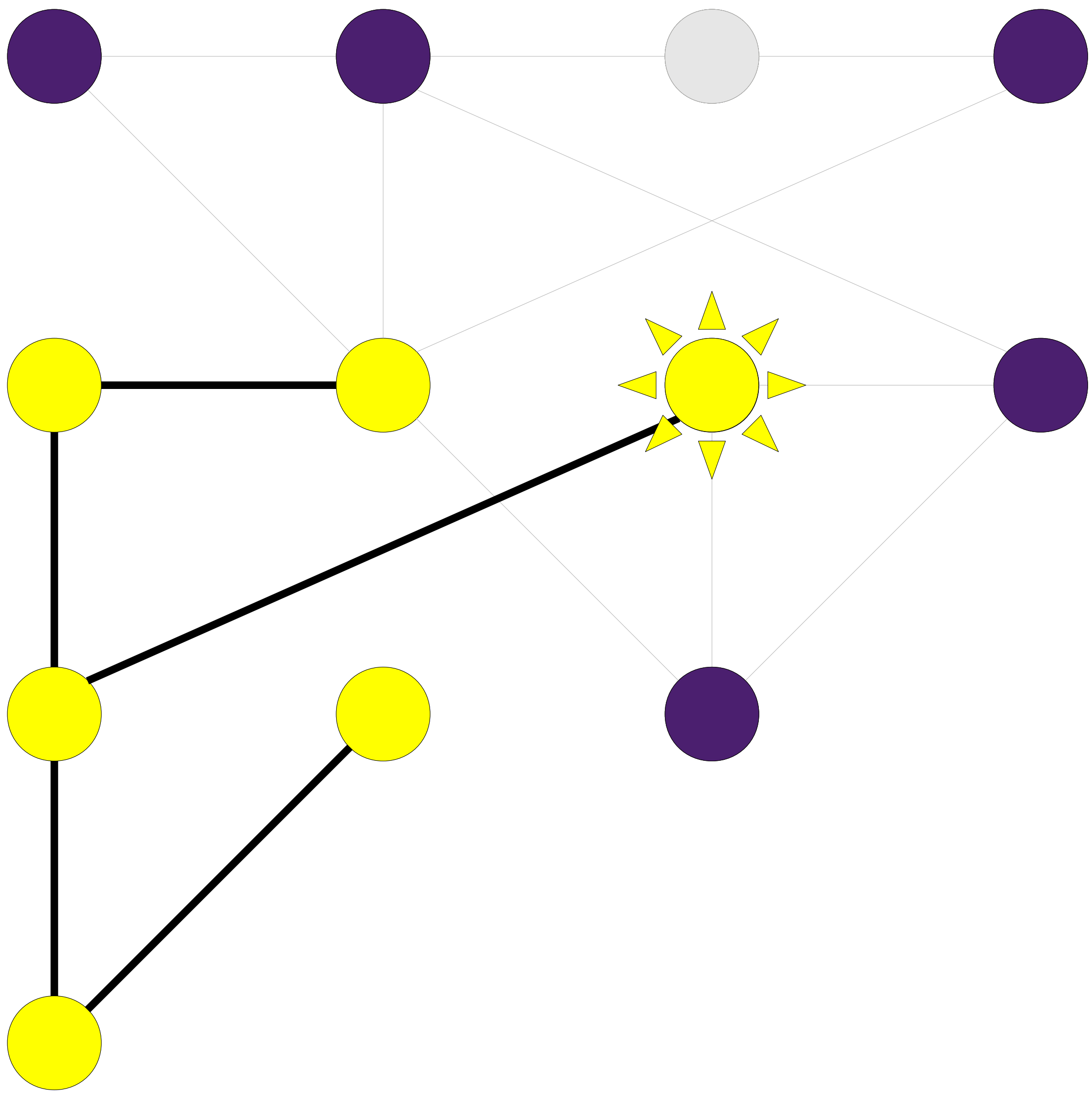
Recursive Depth-First Search



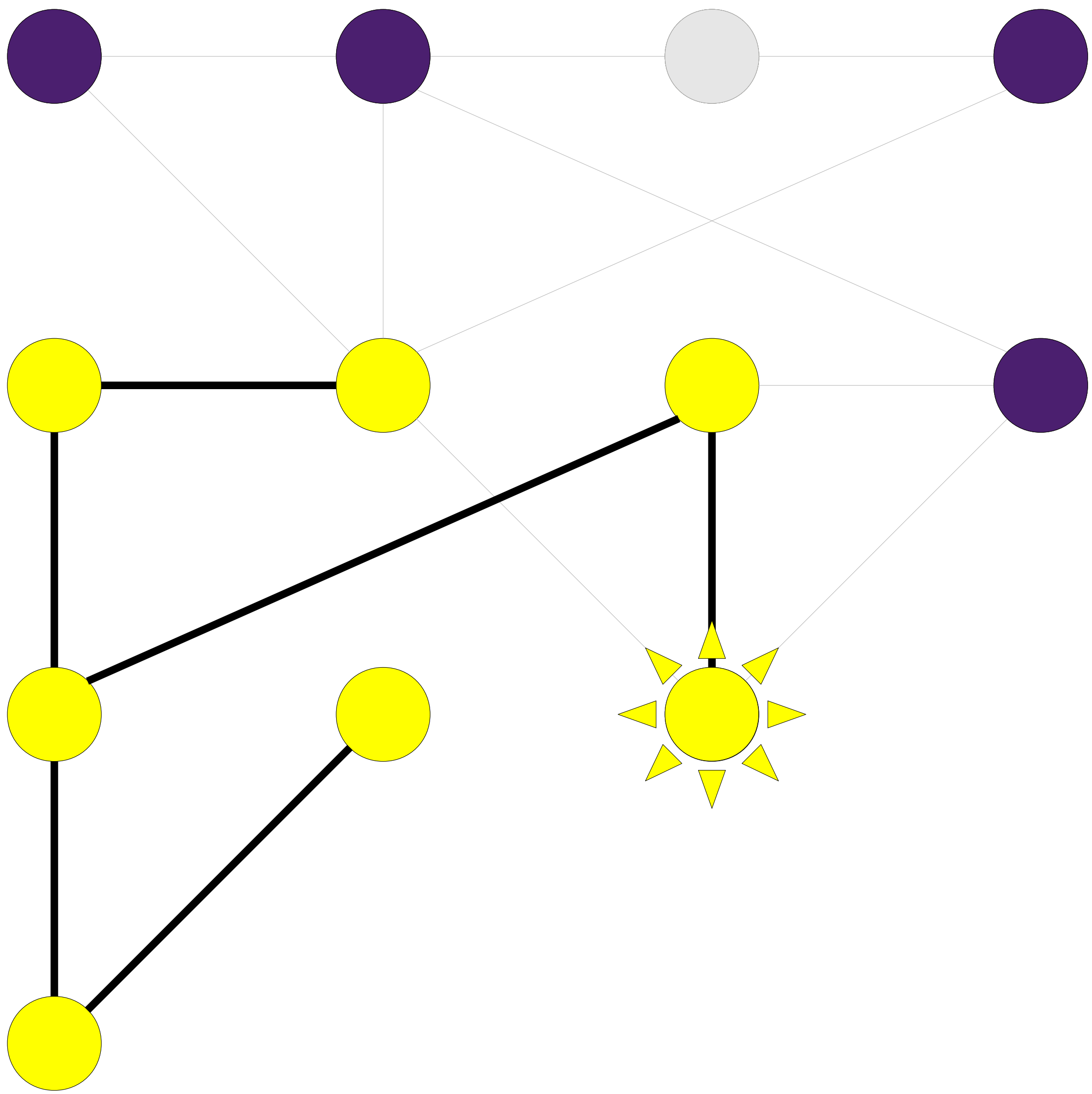
Recursive Depth-First Search



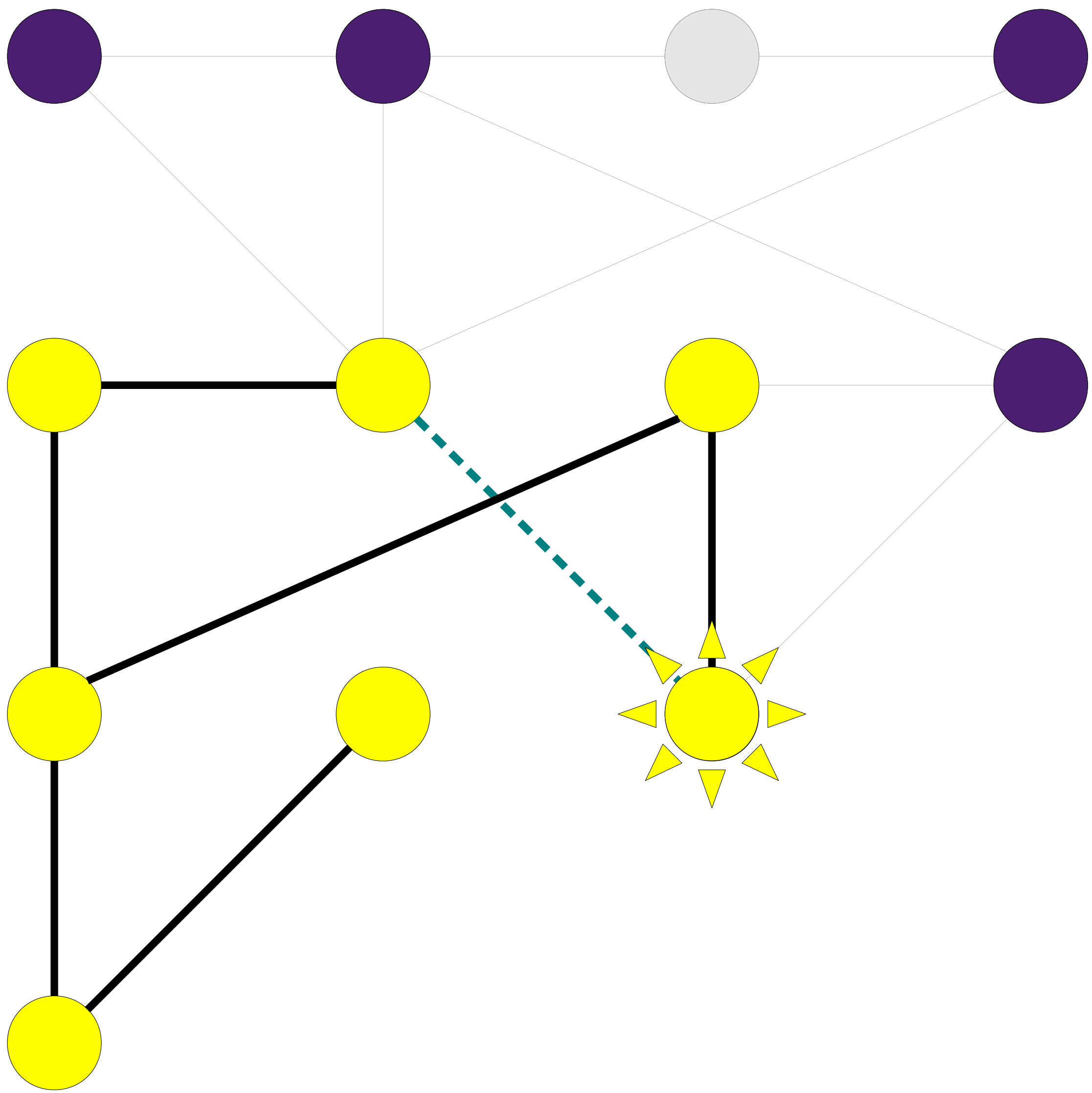
Recursive Depth-First Search



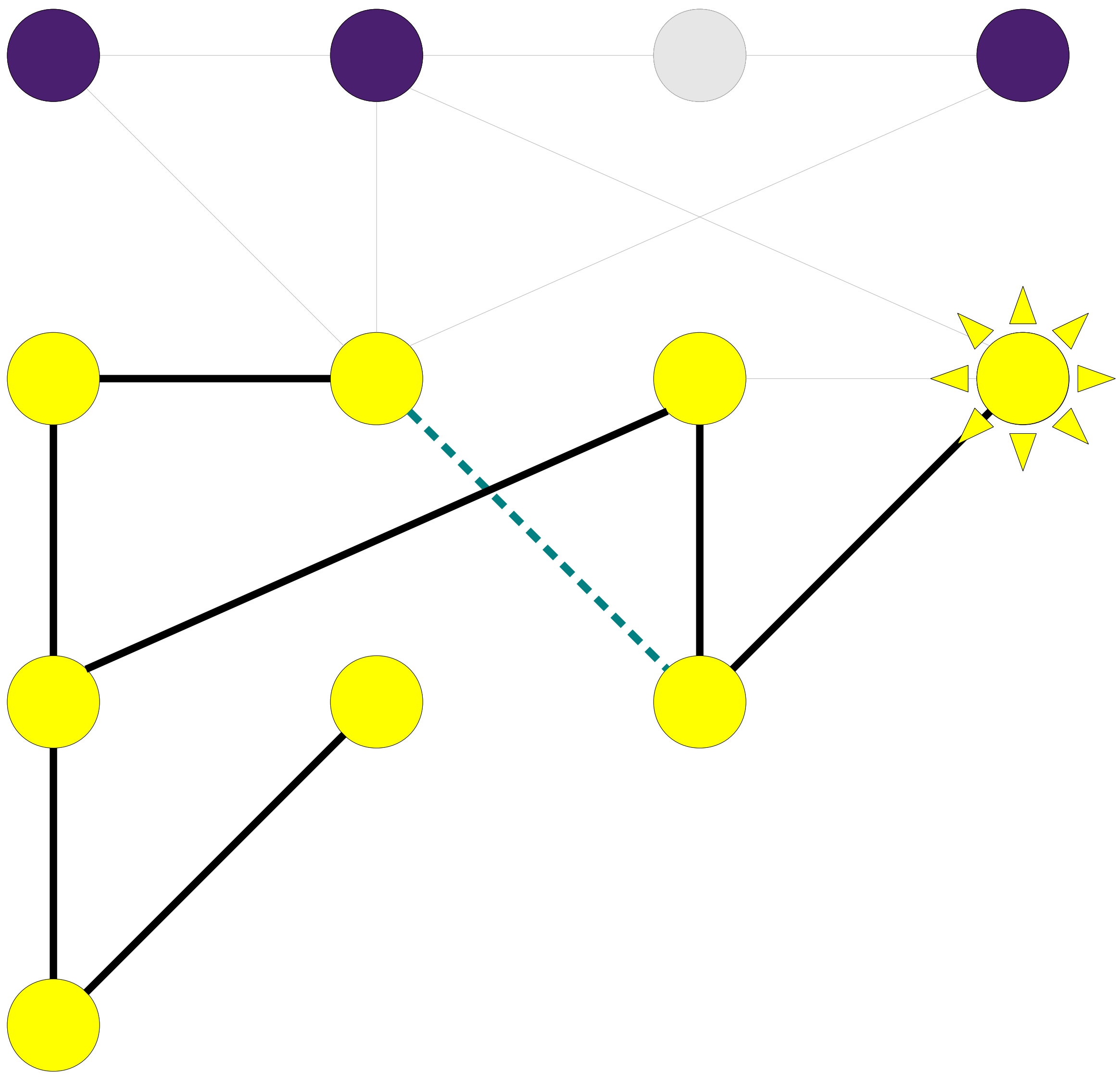
Recursive Depth-First Search



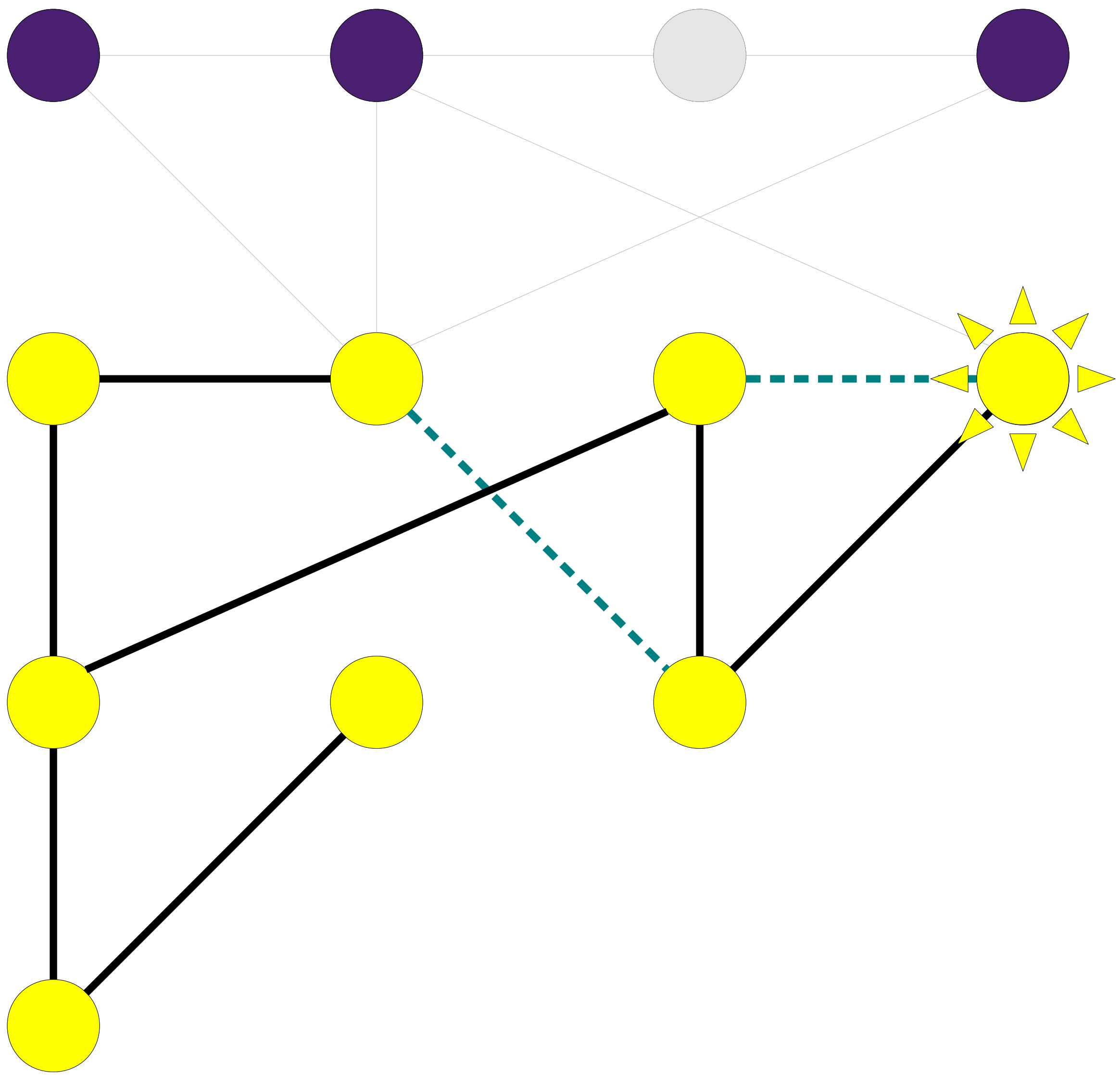
Recursive Depth-First Search



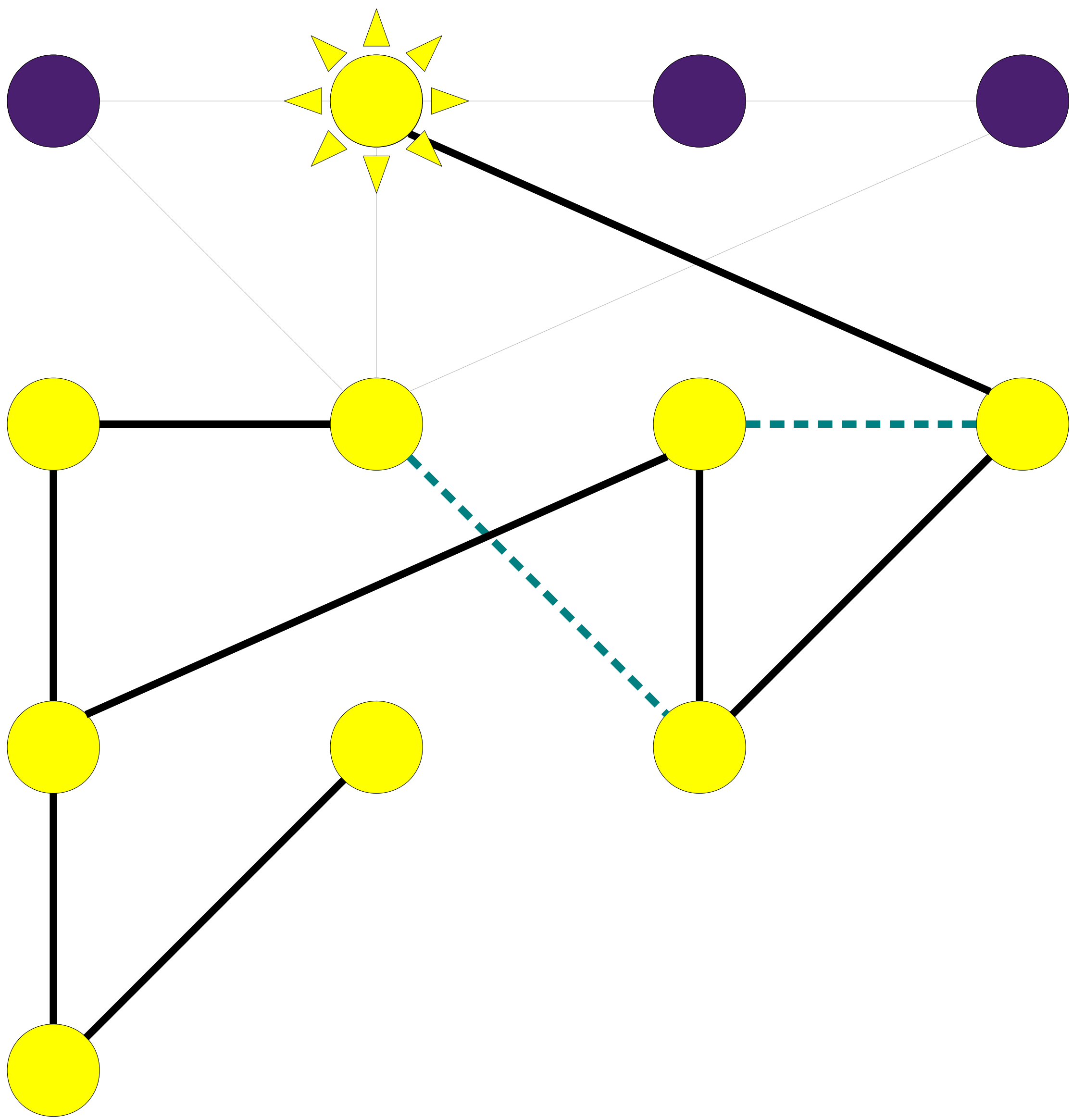
Recursive Depth-First Search



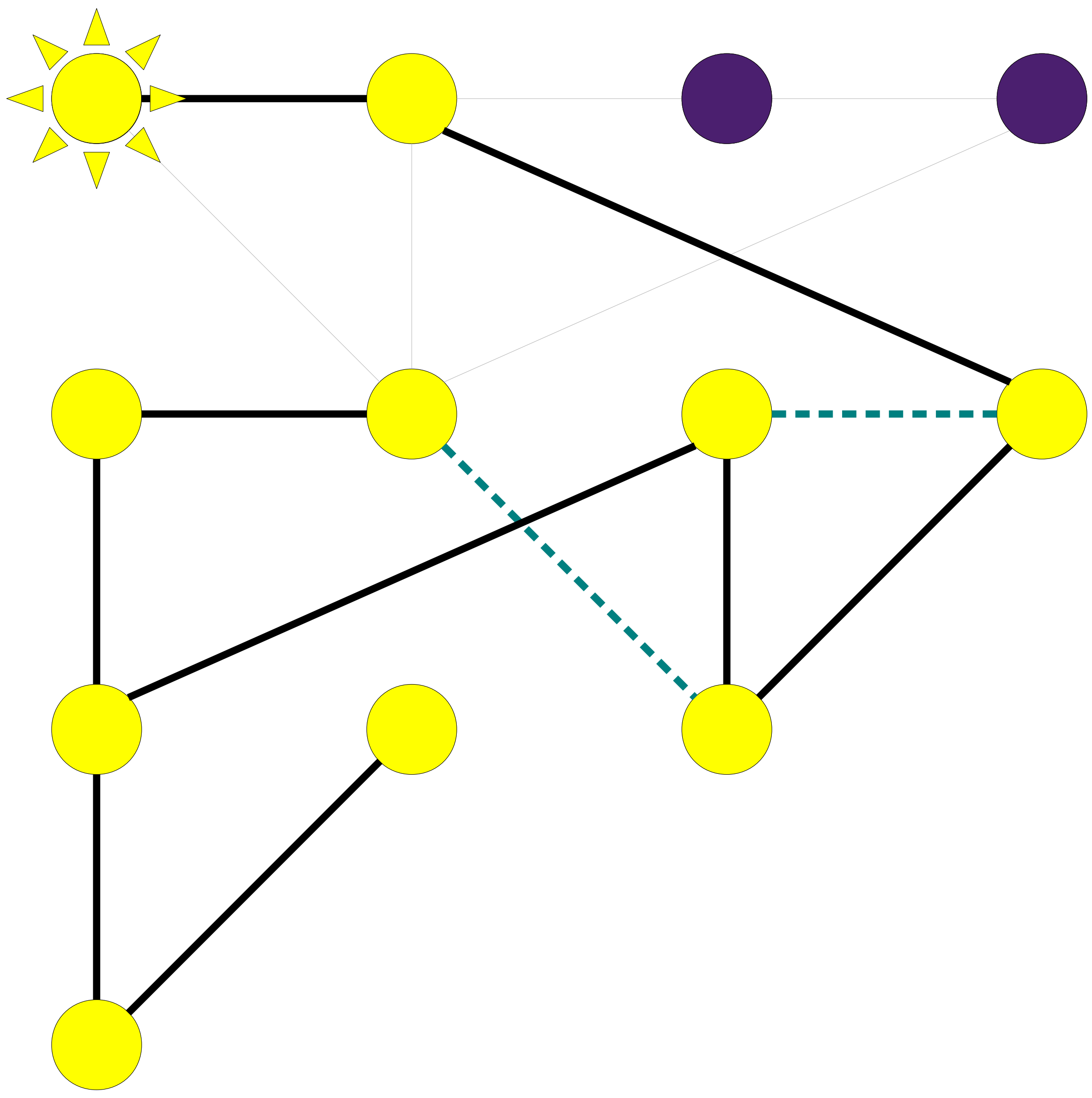
Recursive Depth-First Search



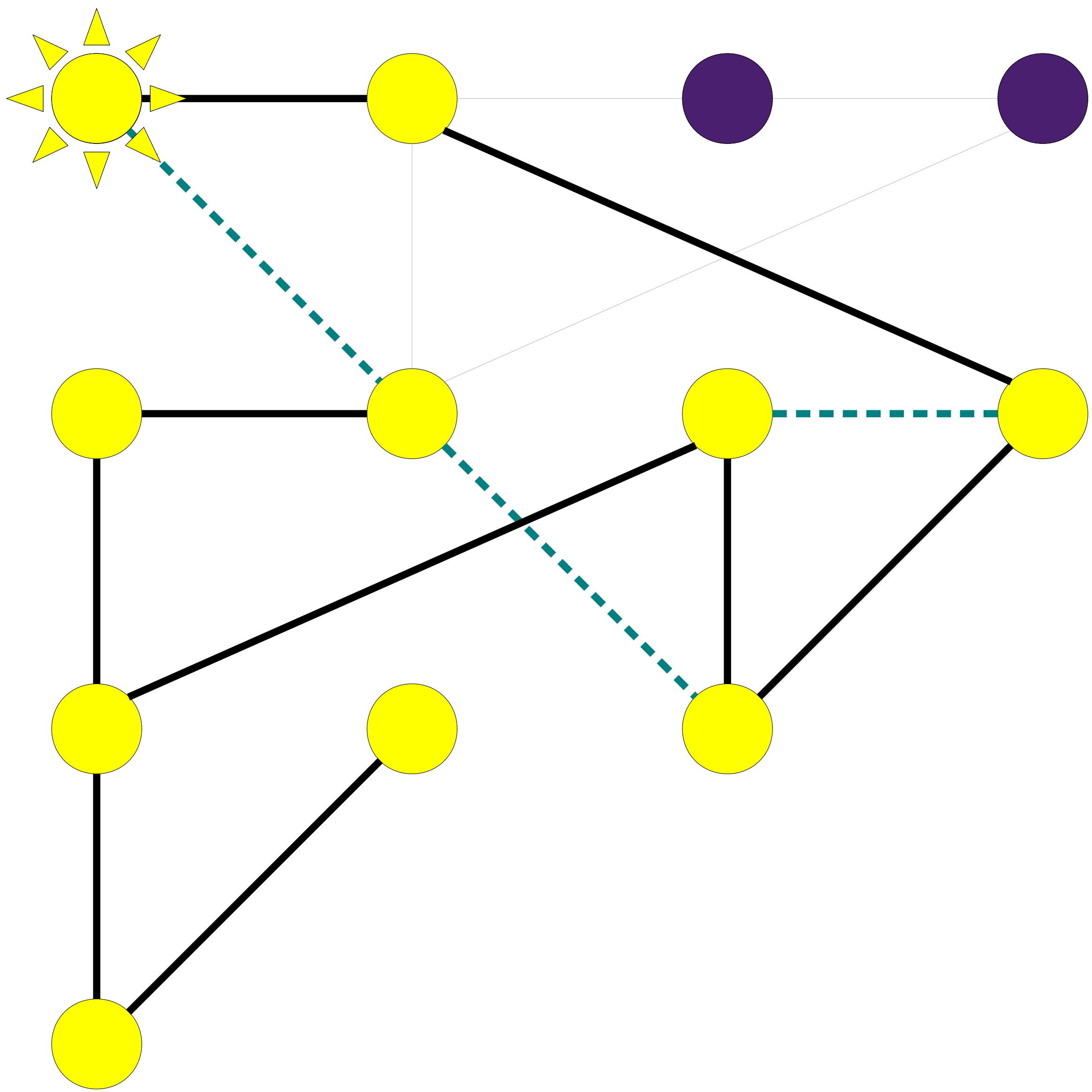
Recursive Depth-First Search



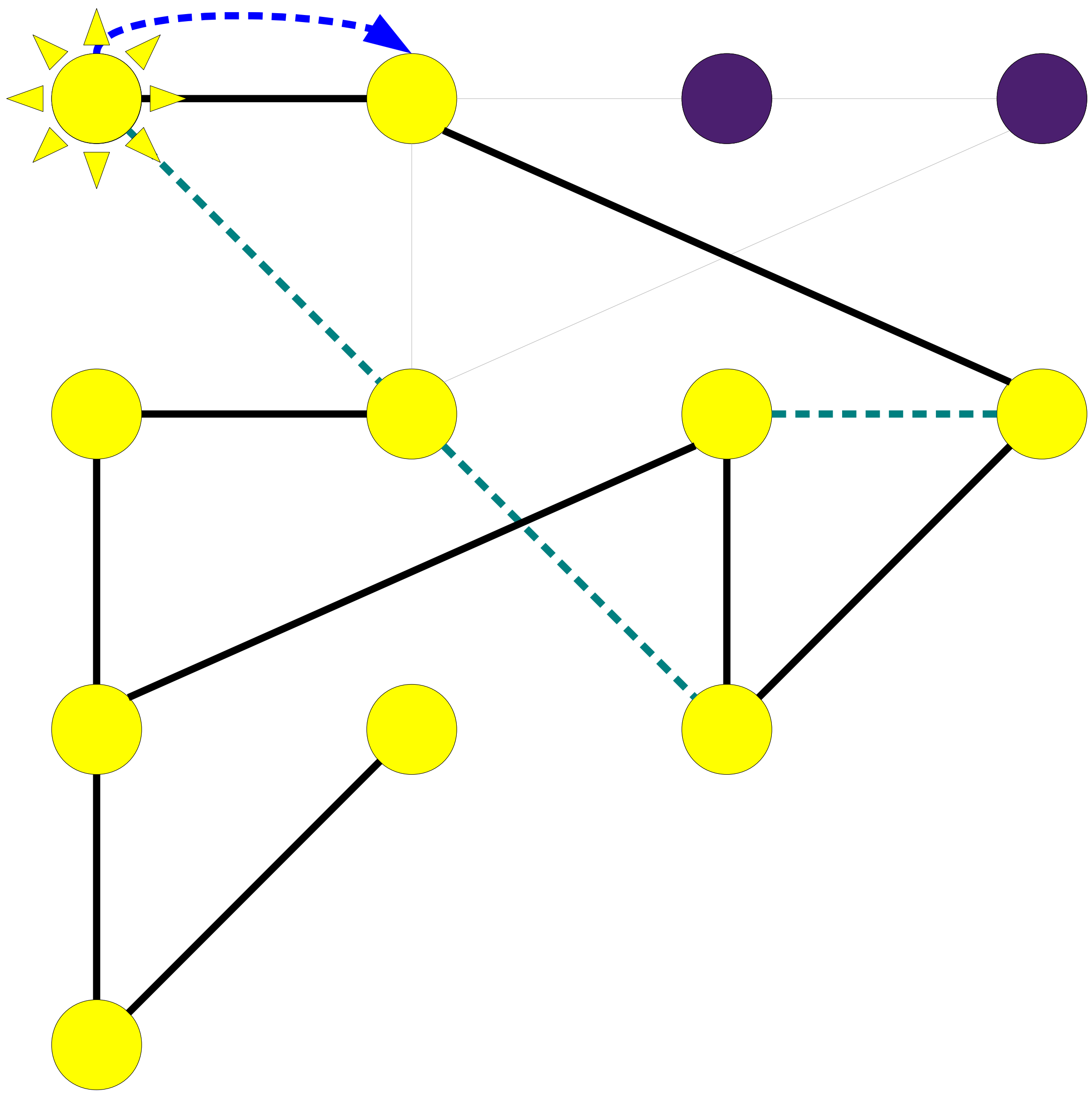
Recursive Depth-First Search



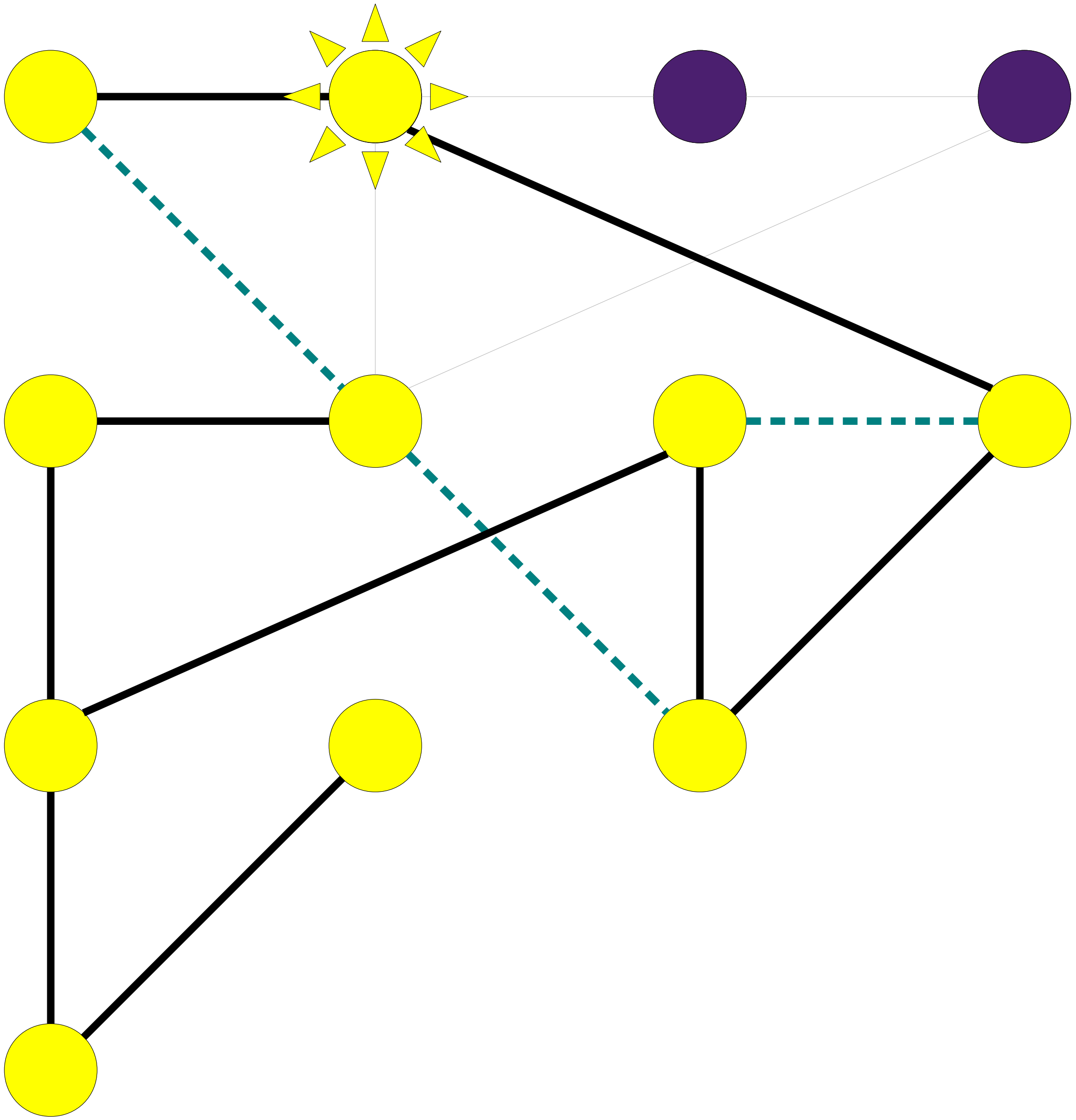
Recursive Depth-First Search



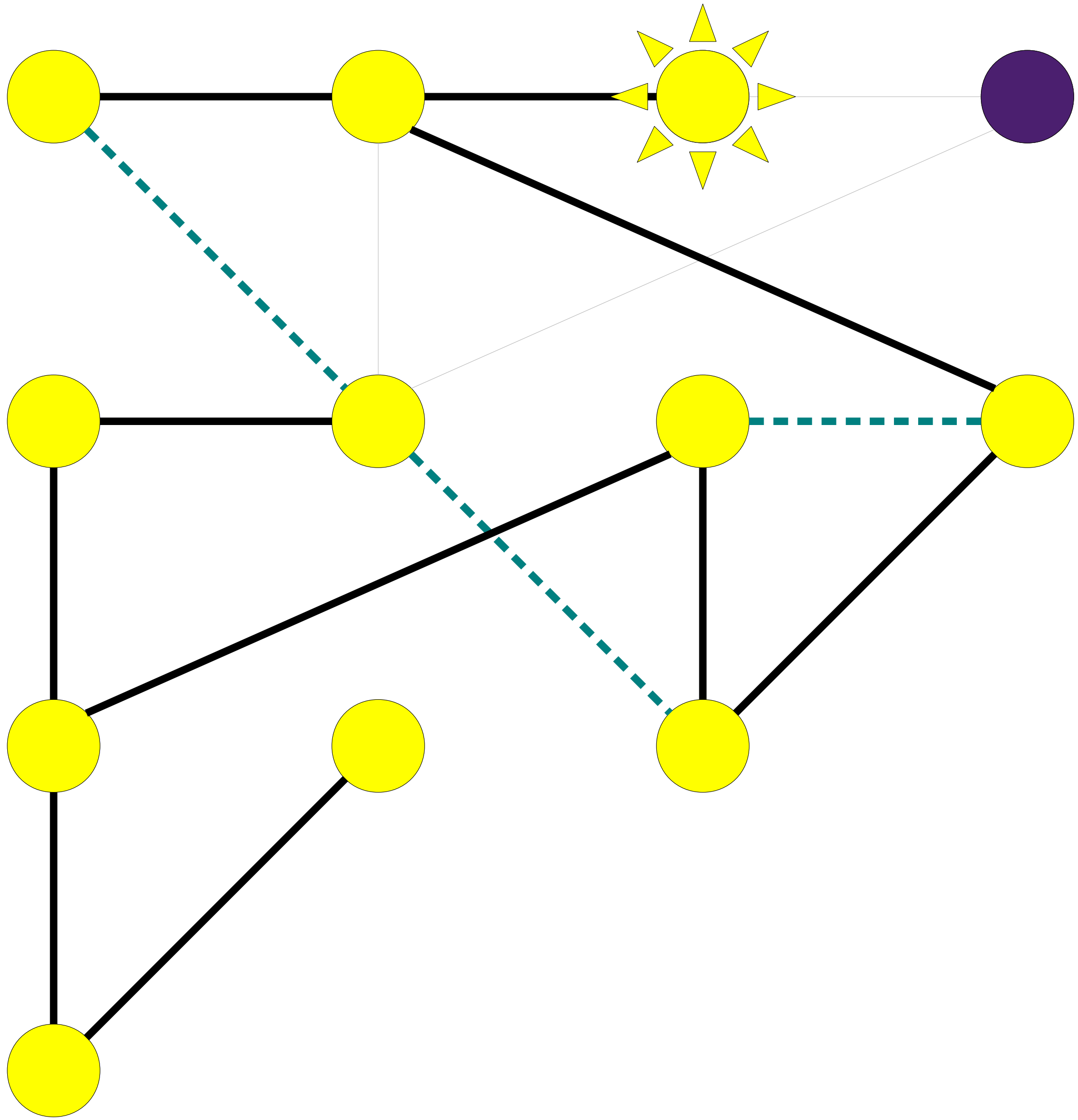
Recursive Depth-First Search



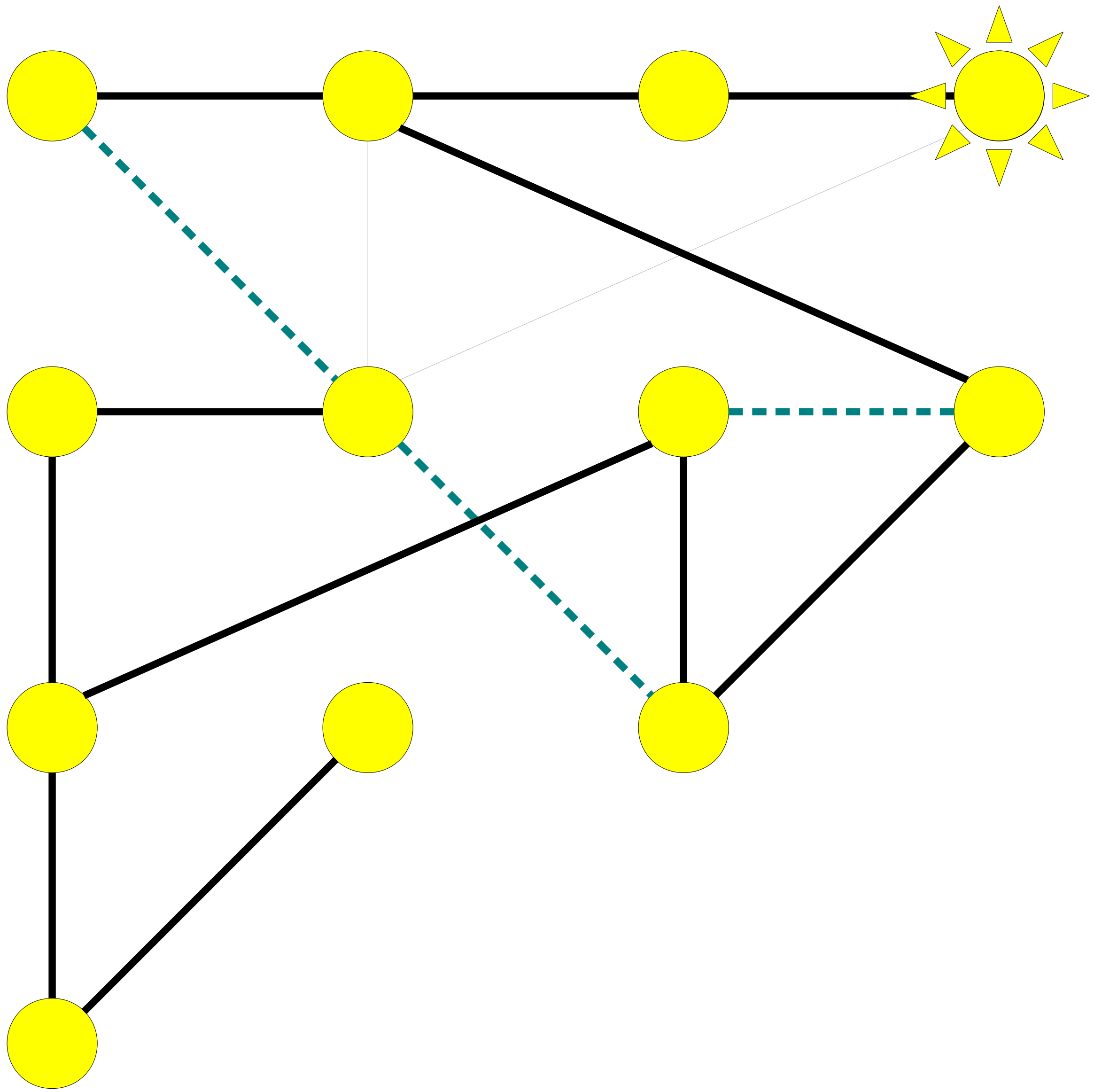
Recursive Depth-First Search



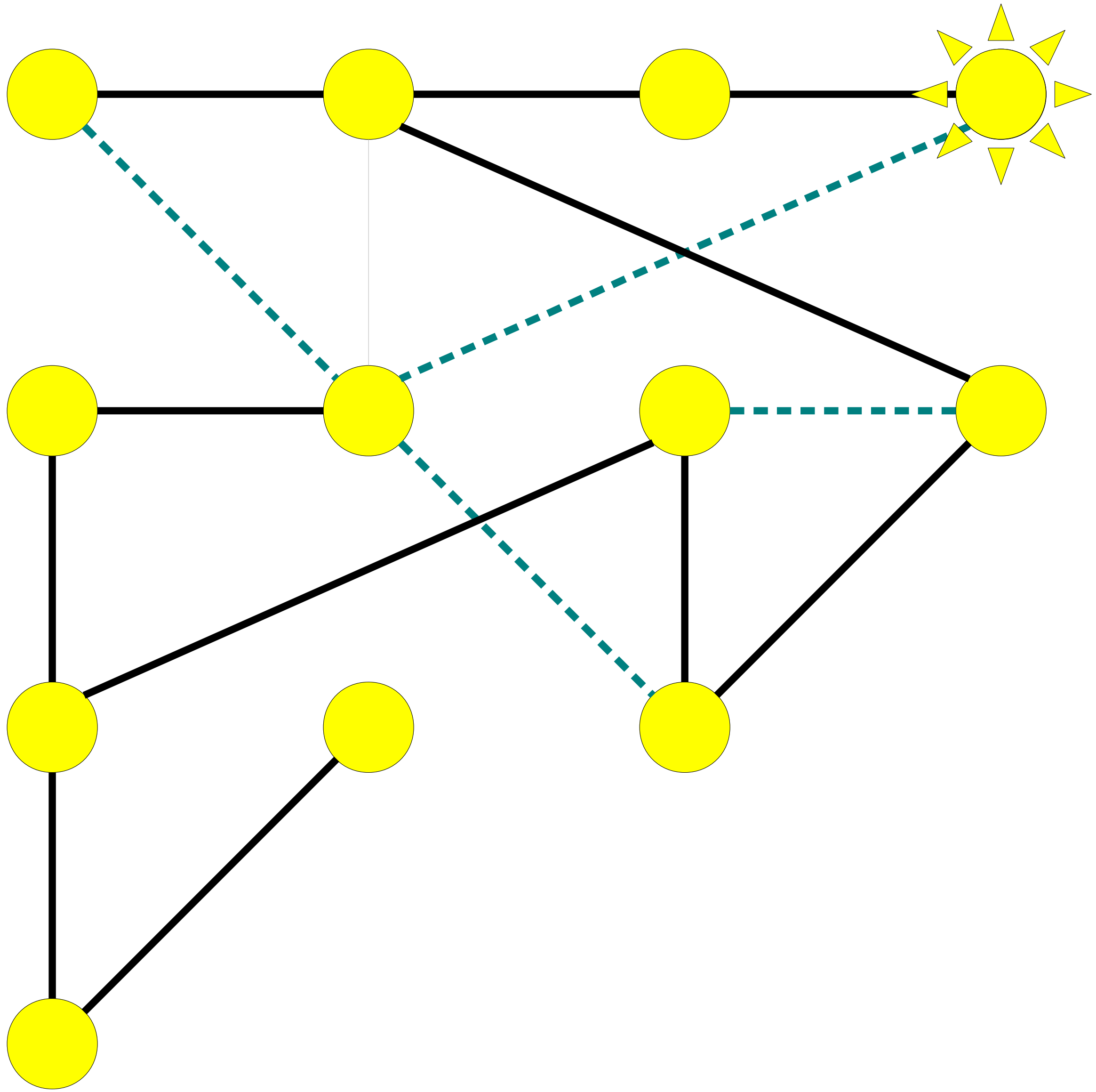
Recursive Depth-First Search



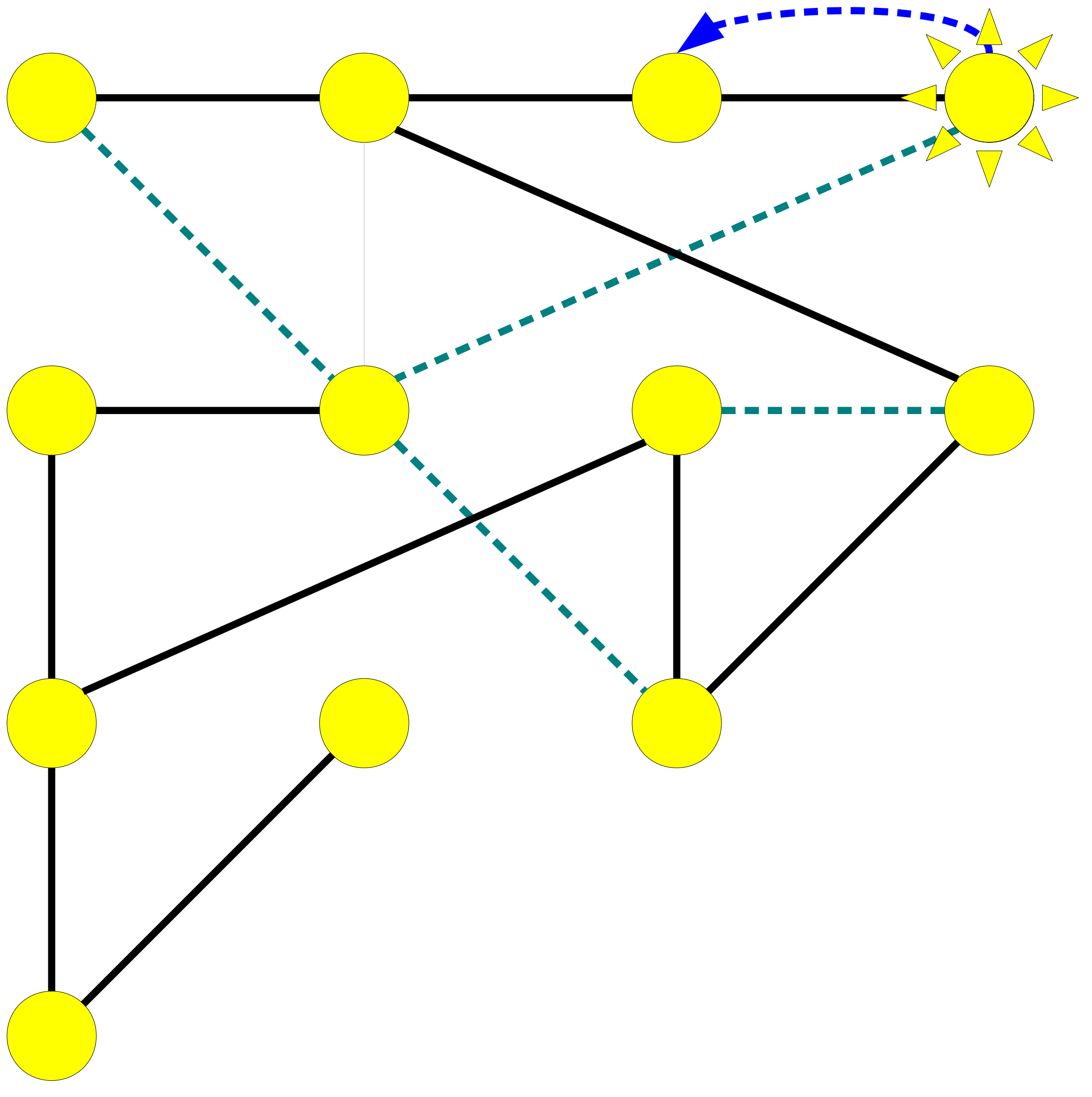
Recursive Depth-First Search



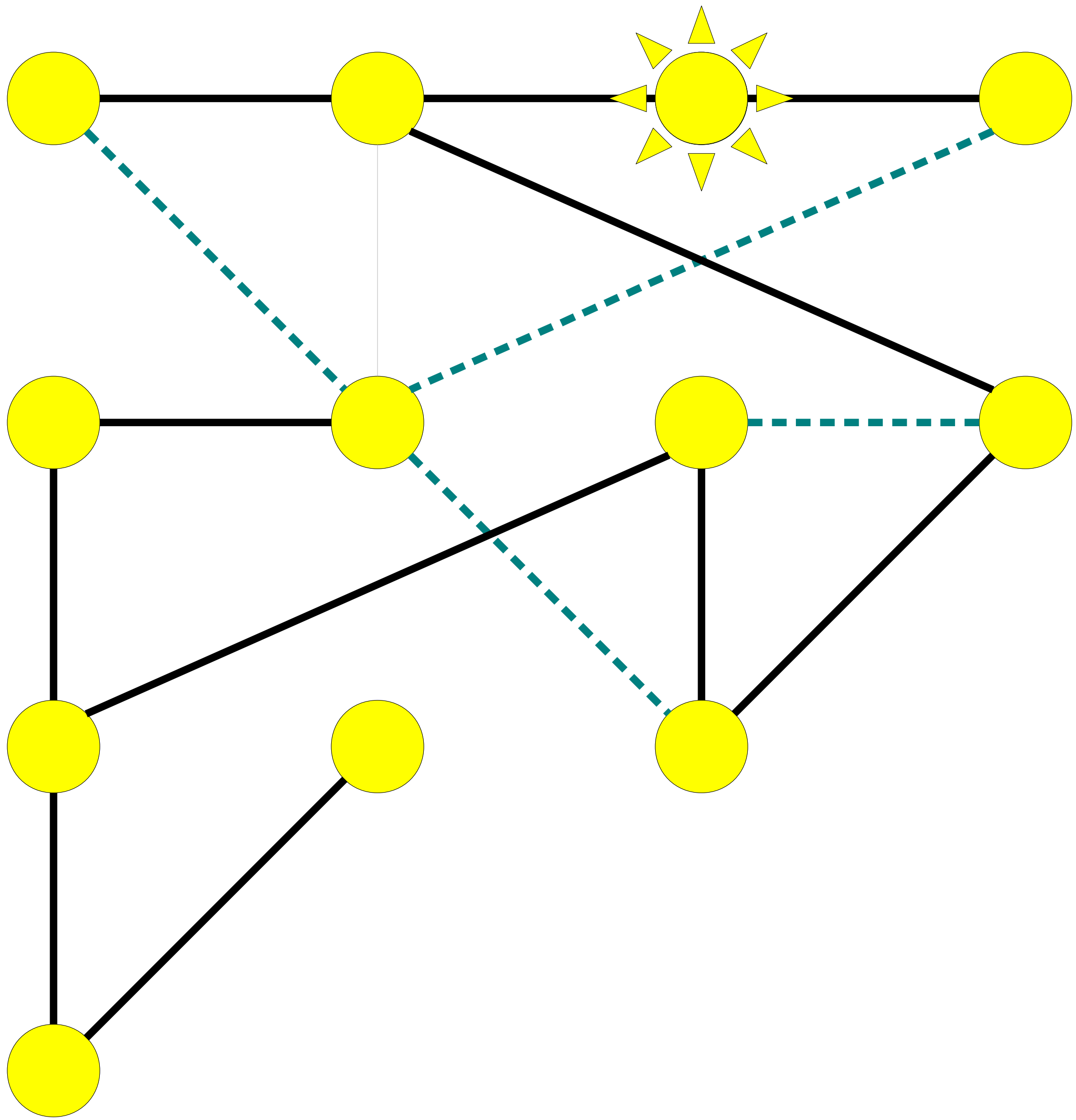
Recursive Depth-First Search



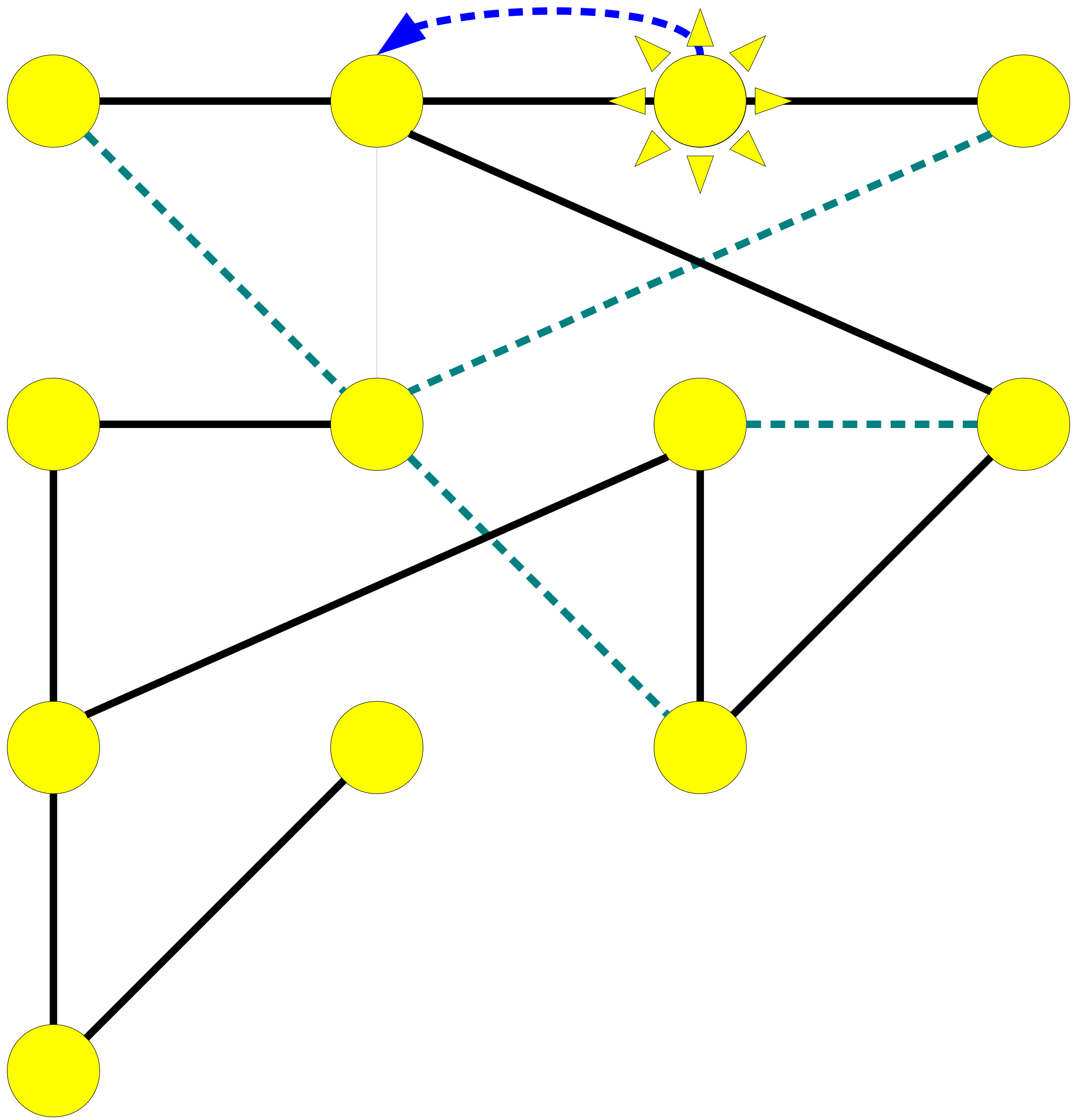
Recursive Depth-First Search



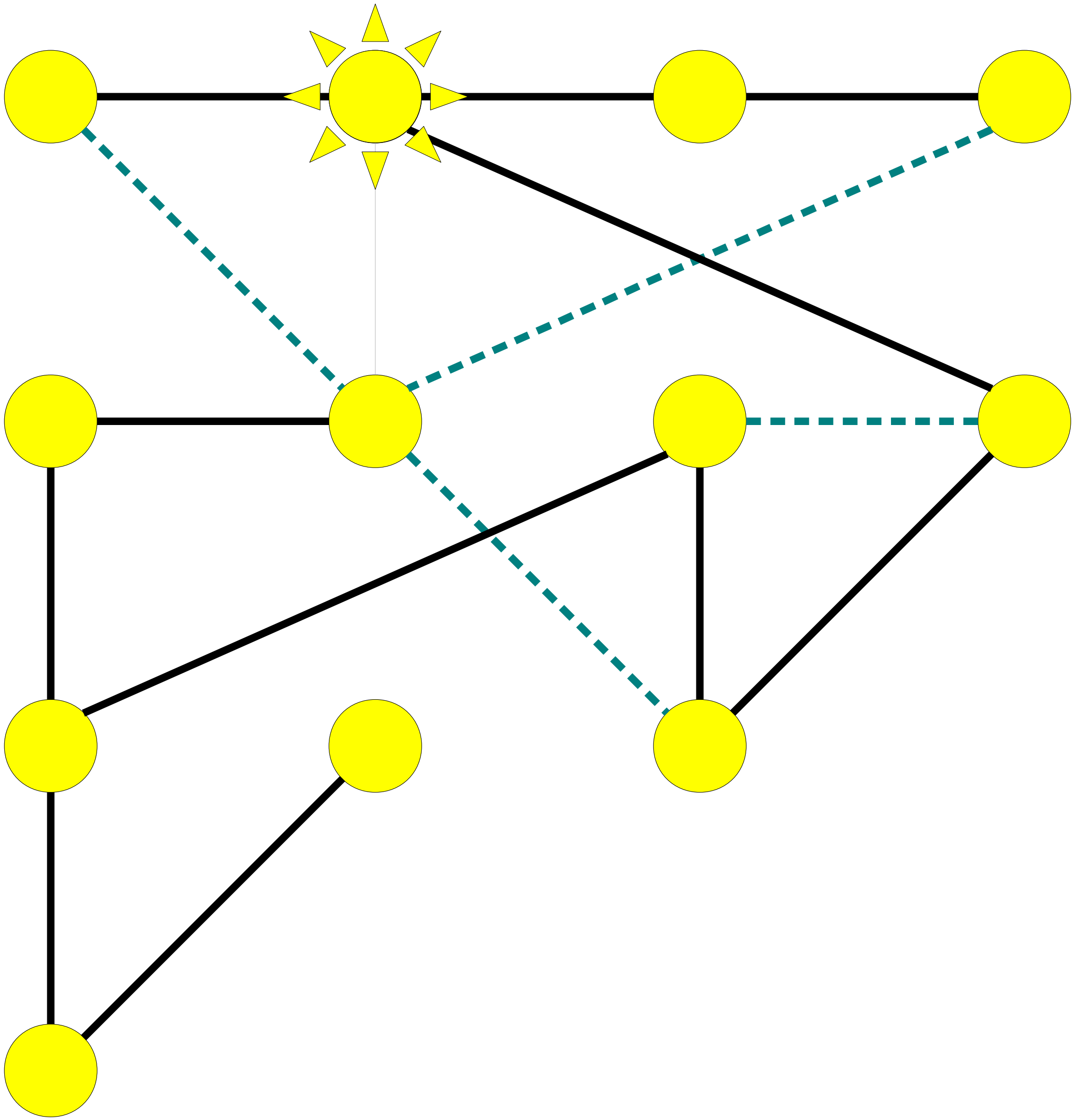
Recursive Depth-First Search



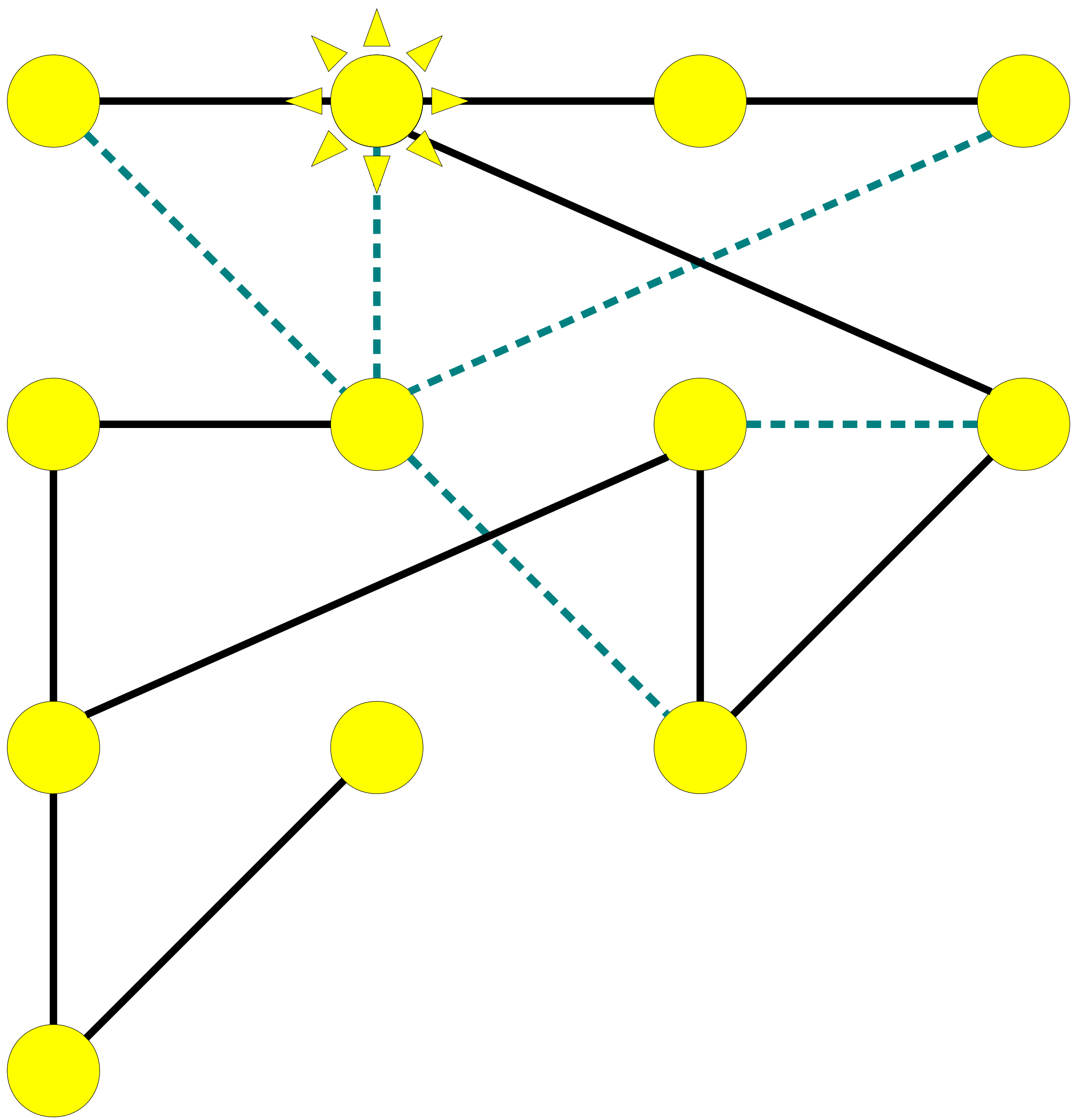
Recursive Depth-First Search



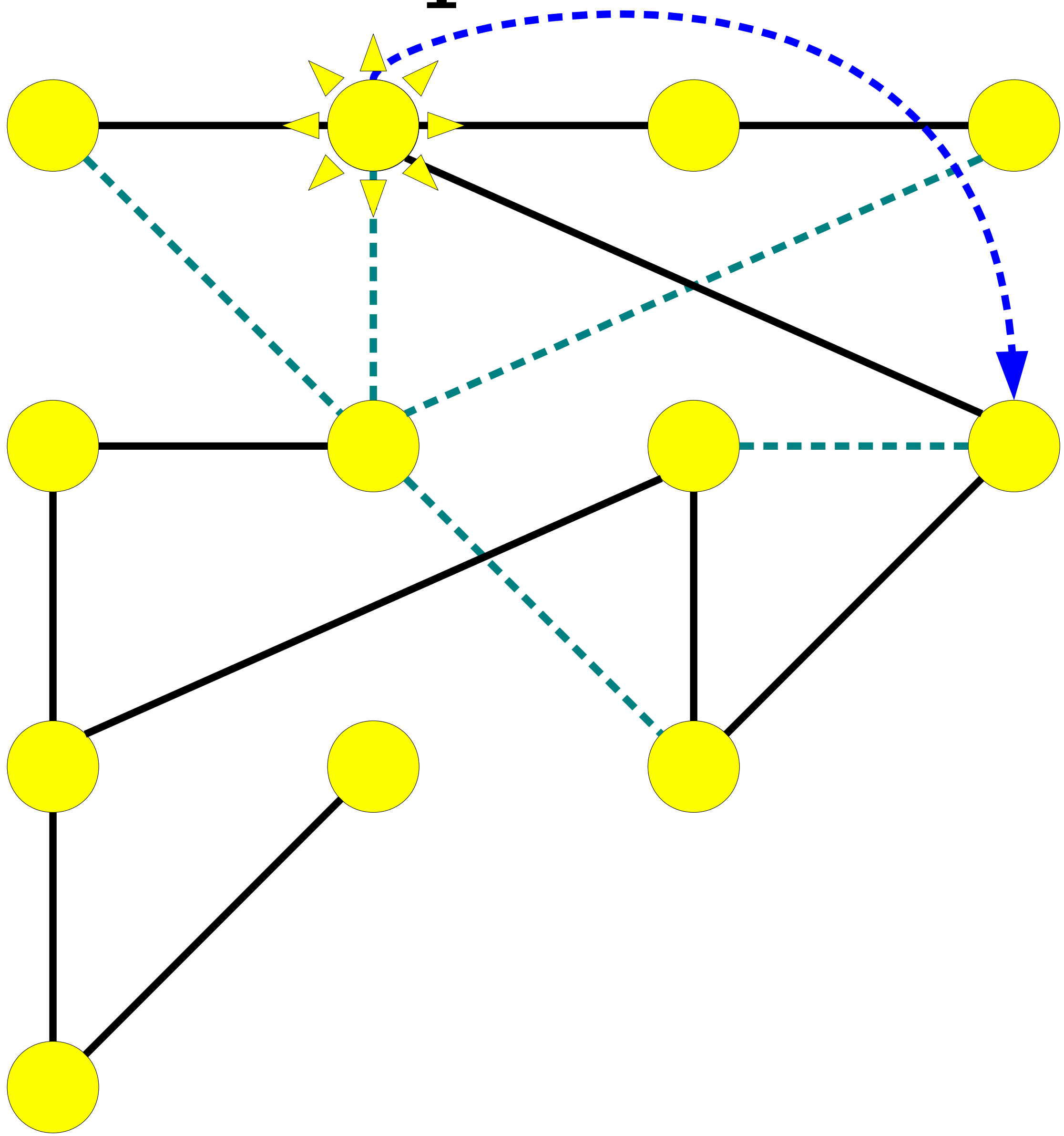
Recursive Depth-First Search



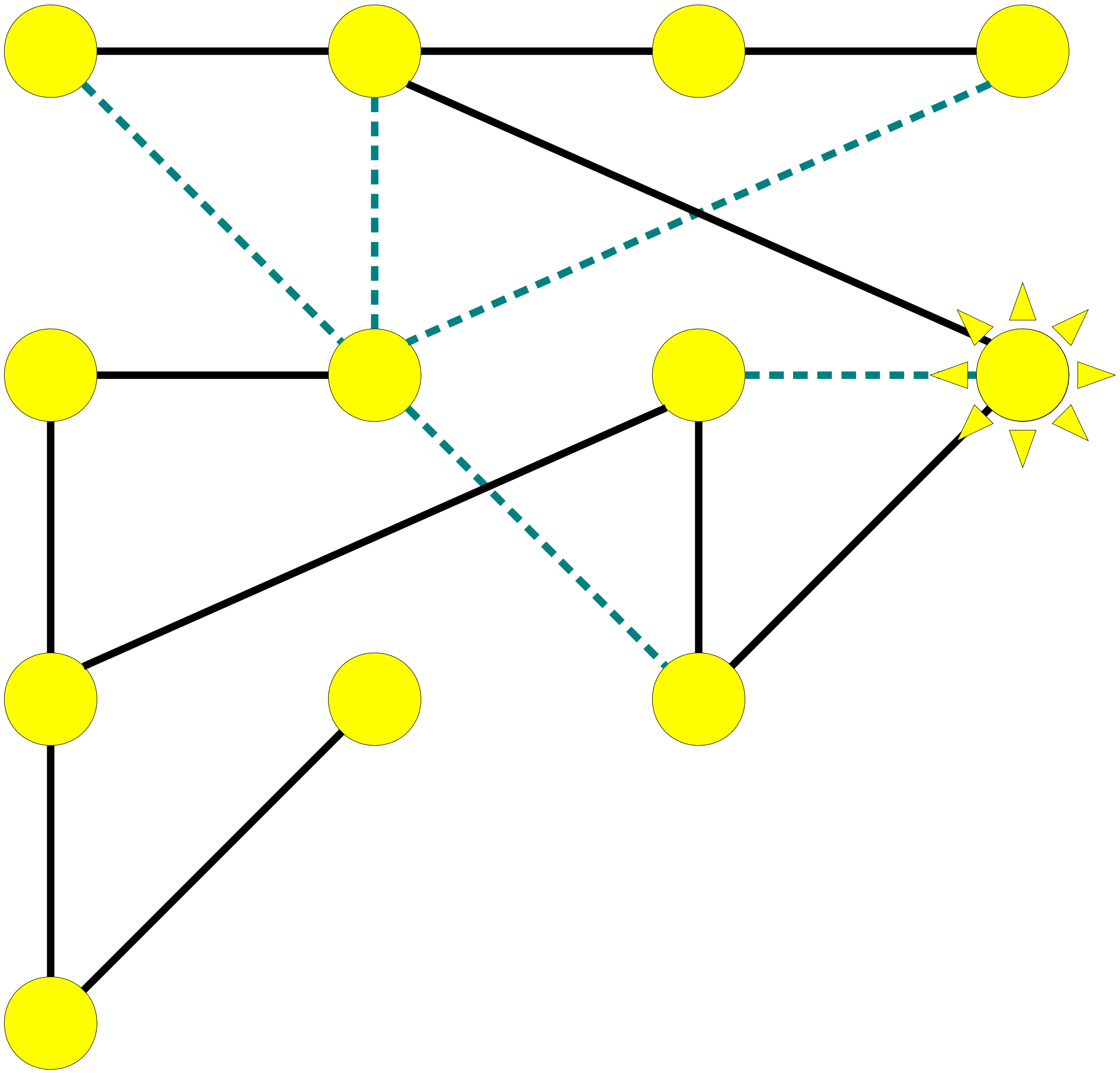
Recursive Depth-First Search



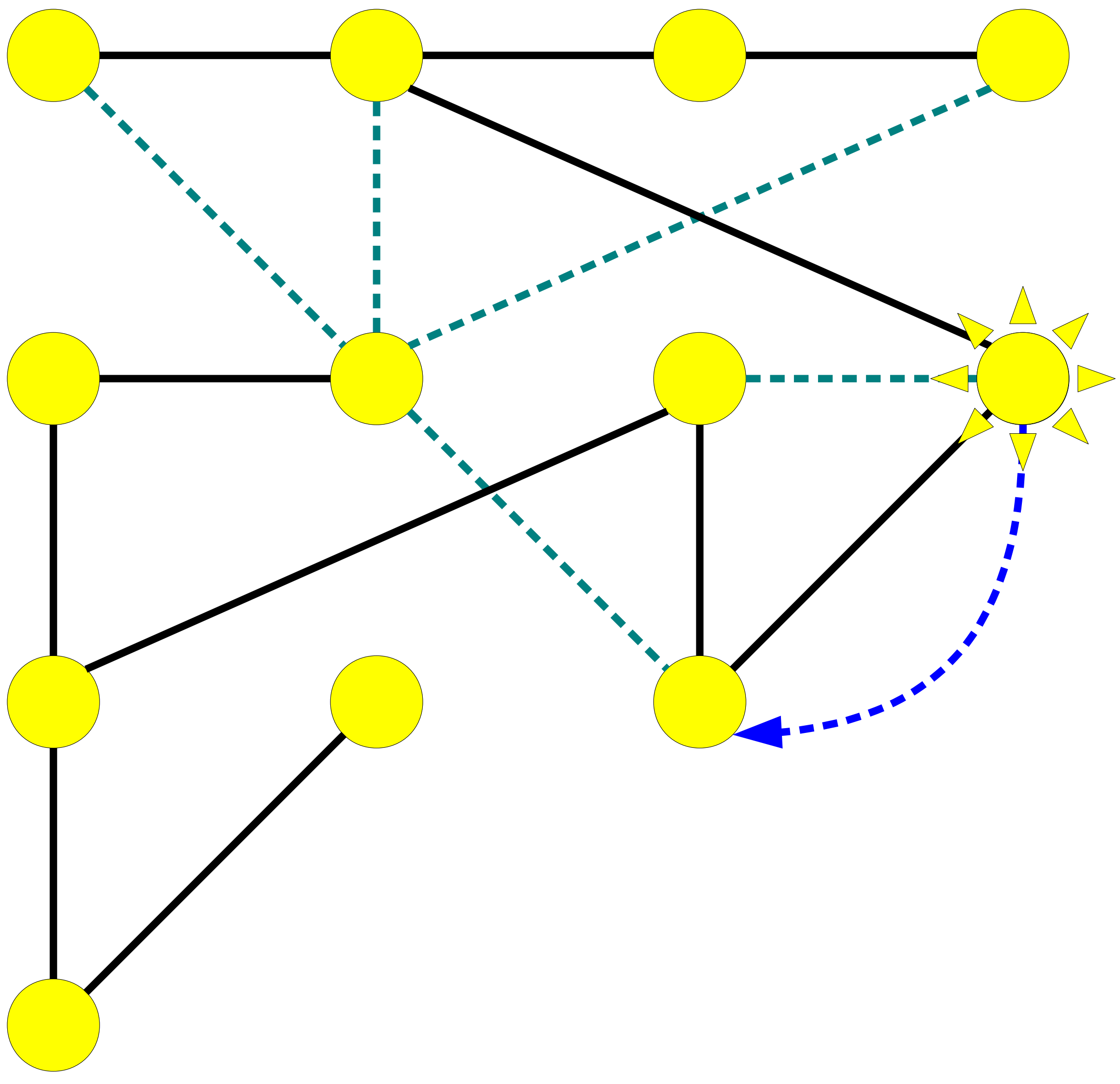
Recursive Depth-First Search



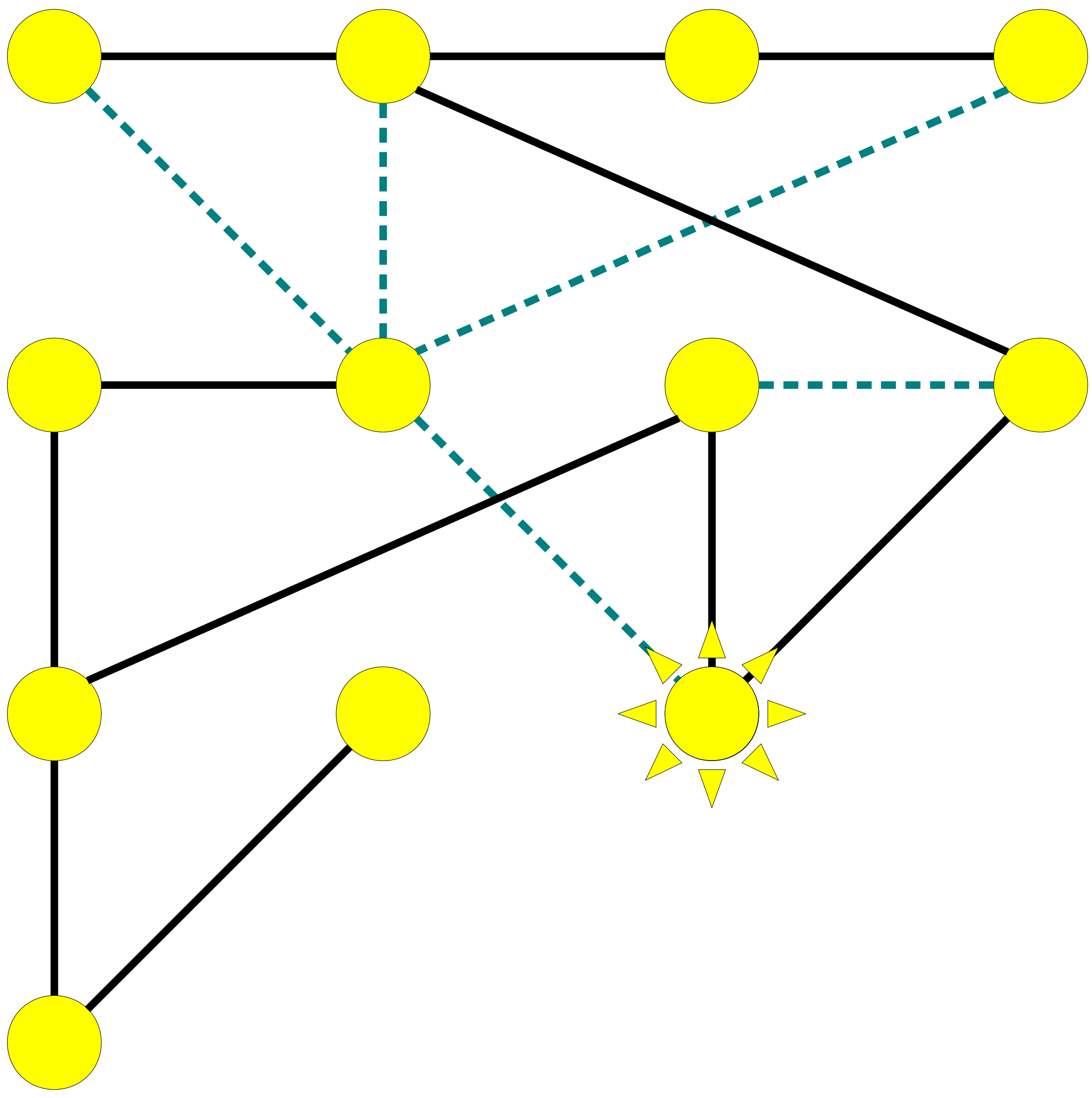
Recursive Depth-First Search



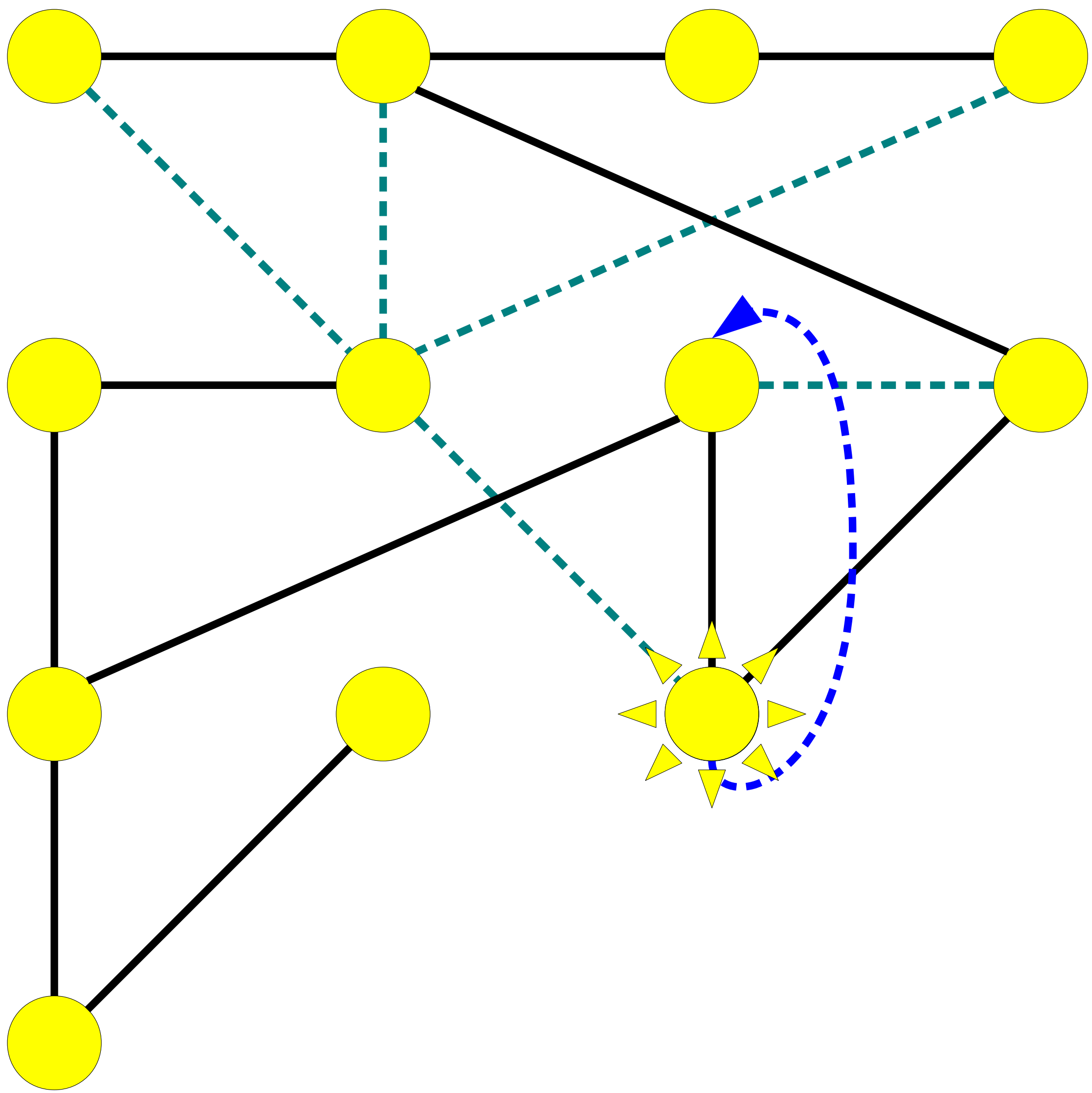
Recursive Depth-First Search



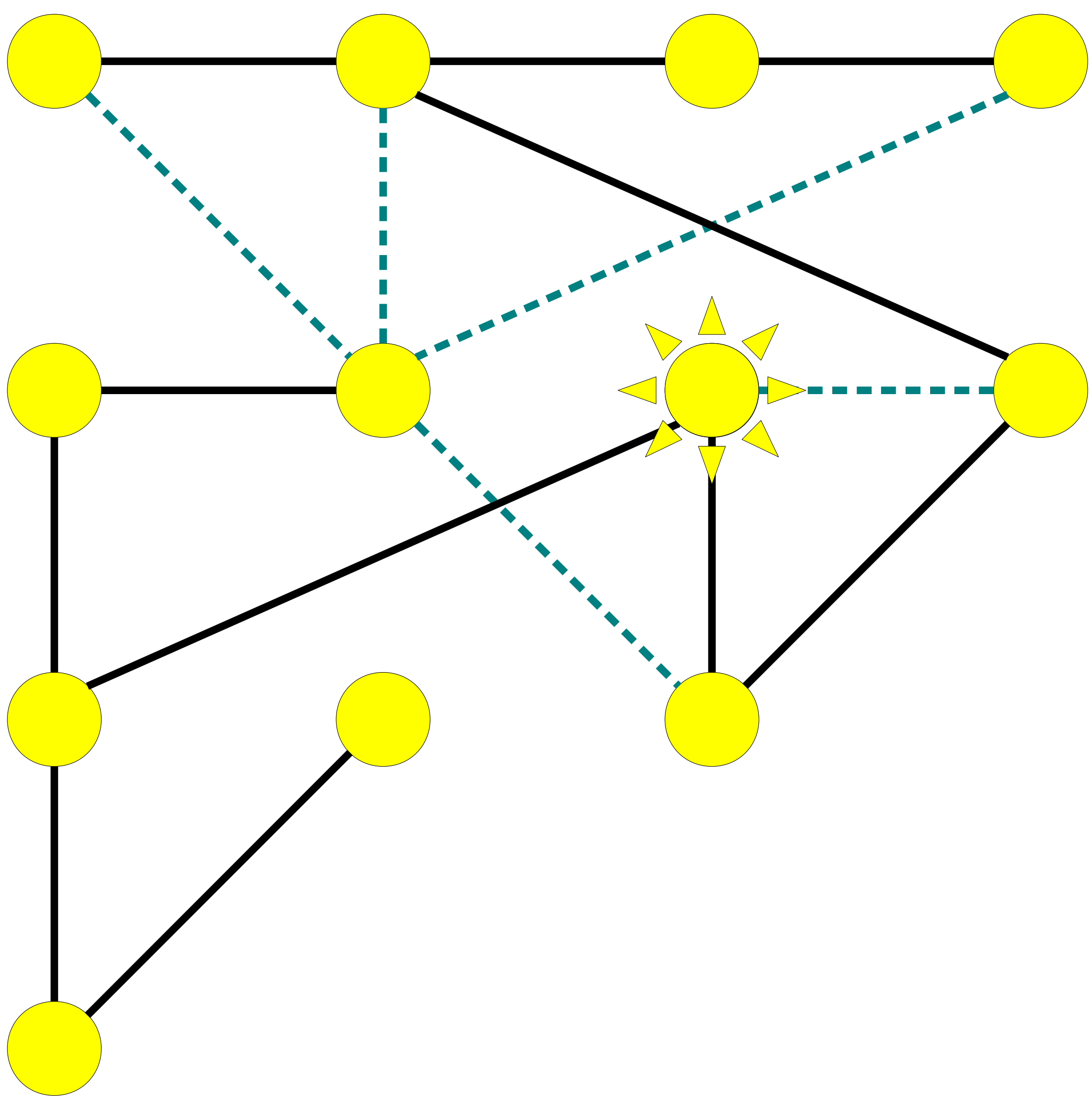
Recursive Depth-First Search



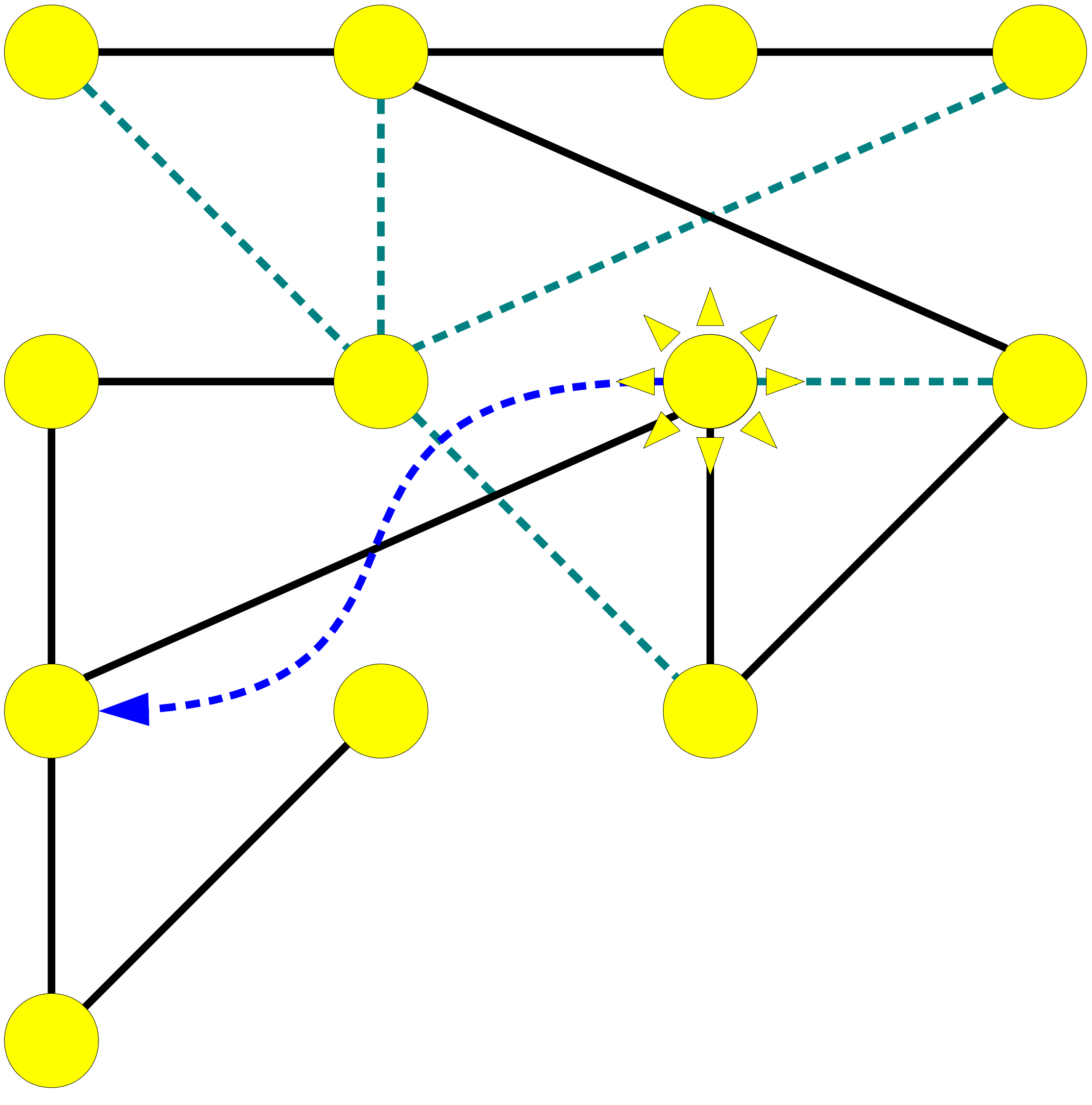
Recursive Depth-First Search



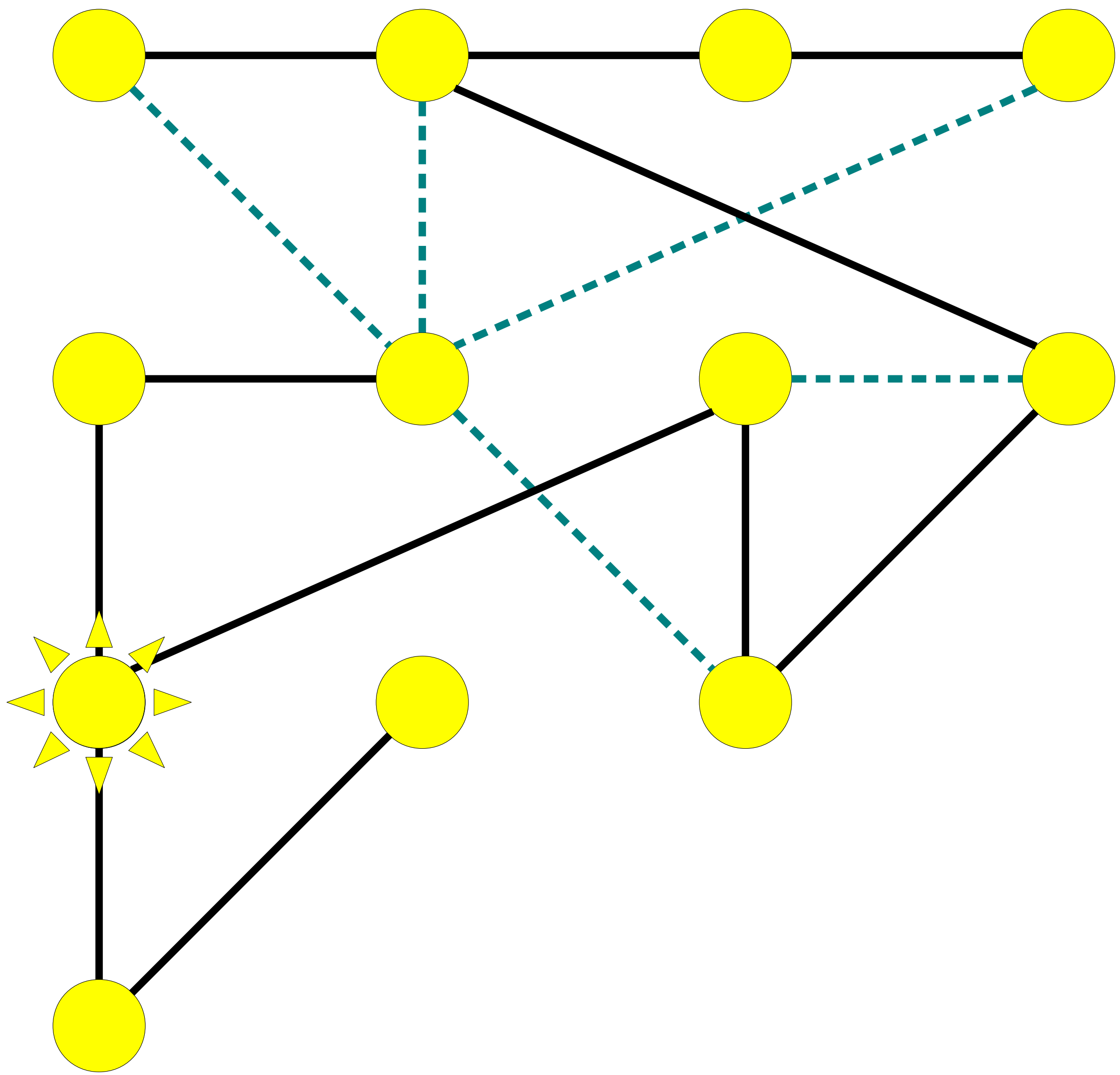
Recursive Depth-First Search



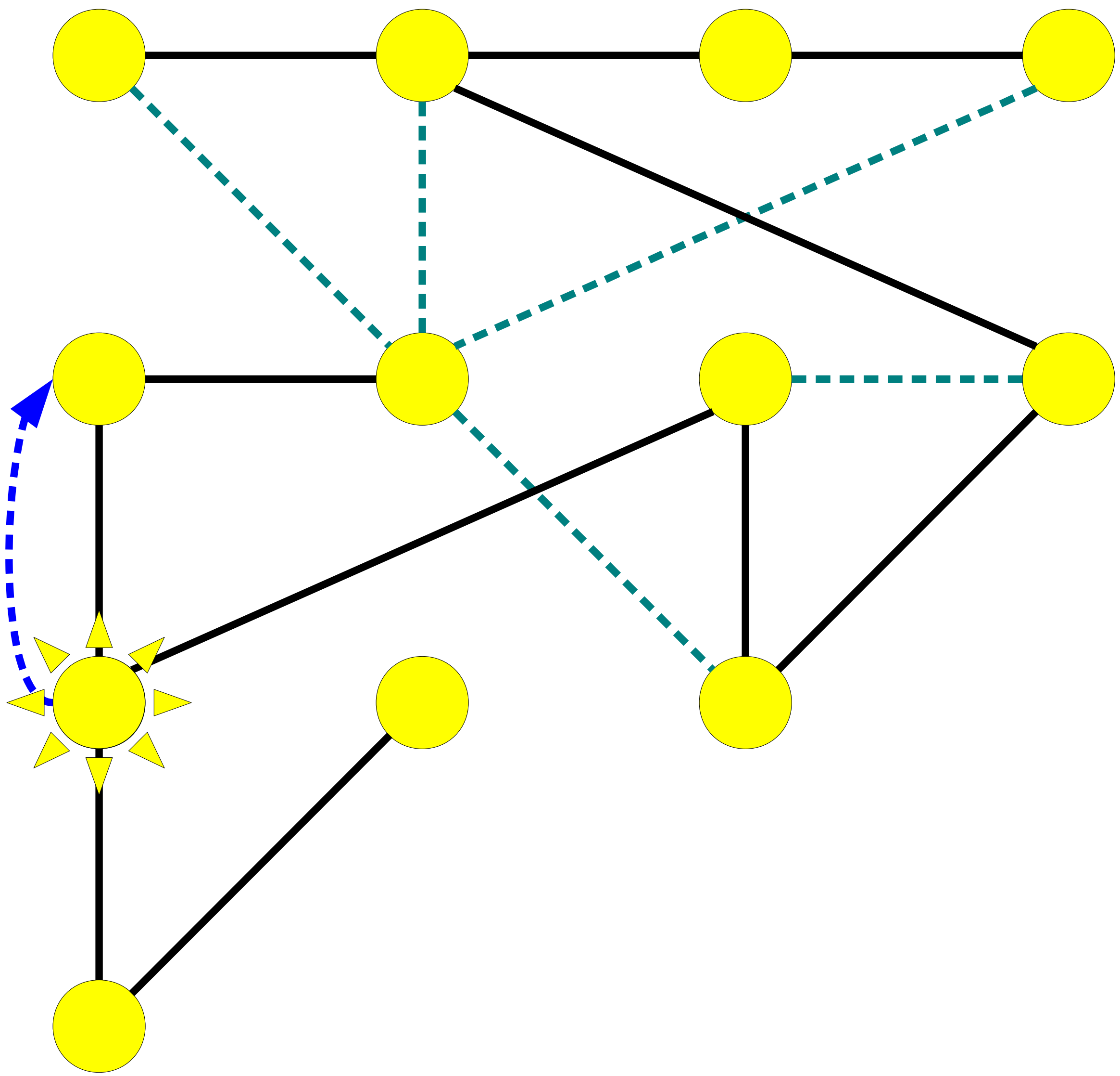
Recursive Depth-First Search



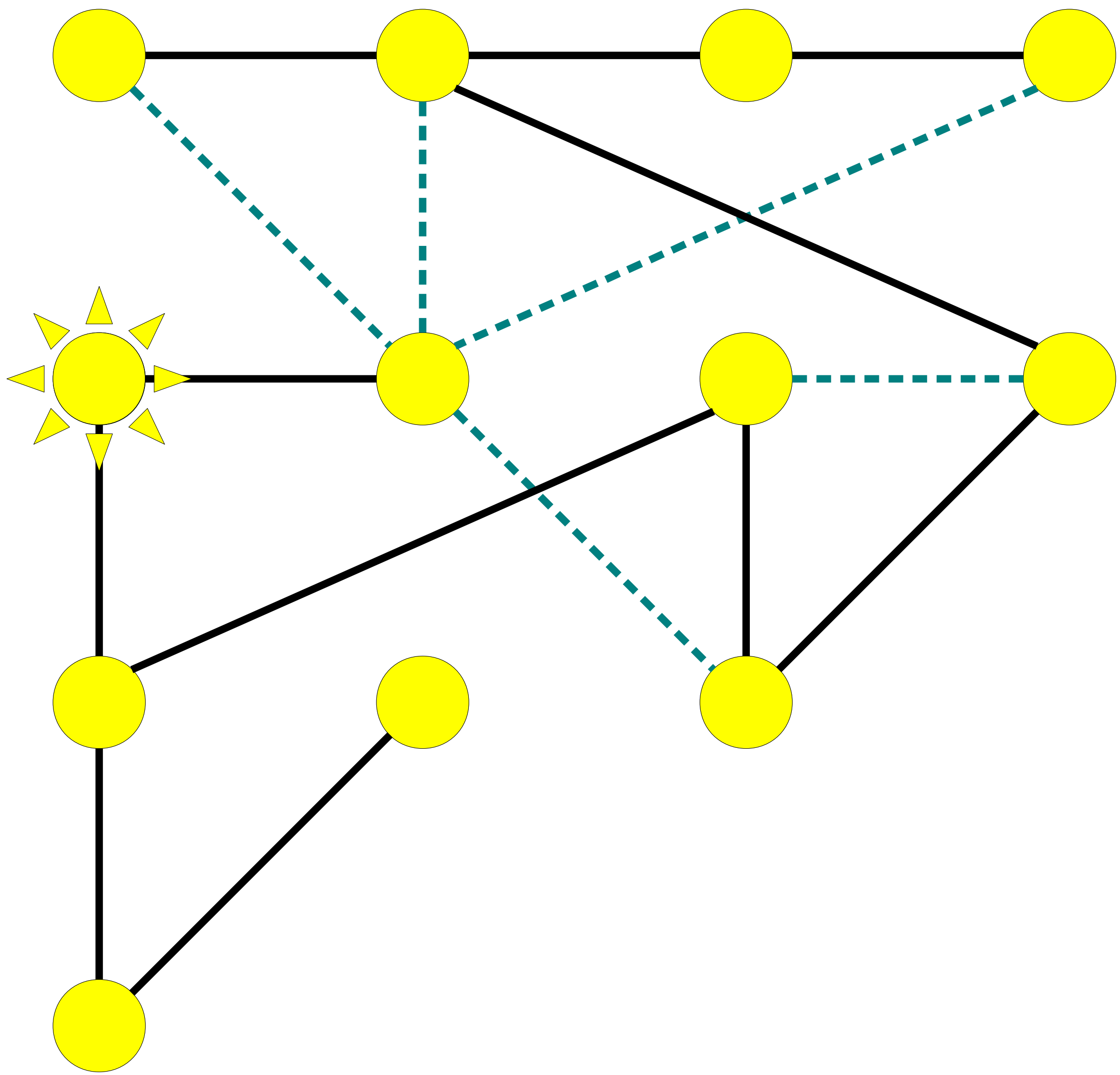
Recursive Depth-First Search



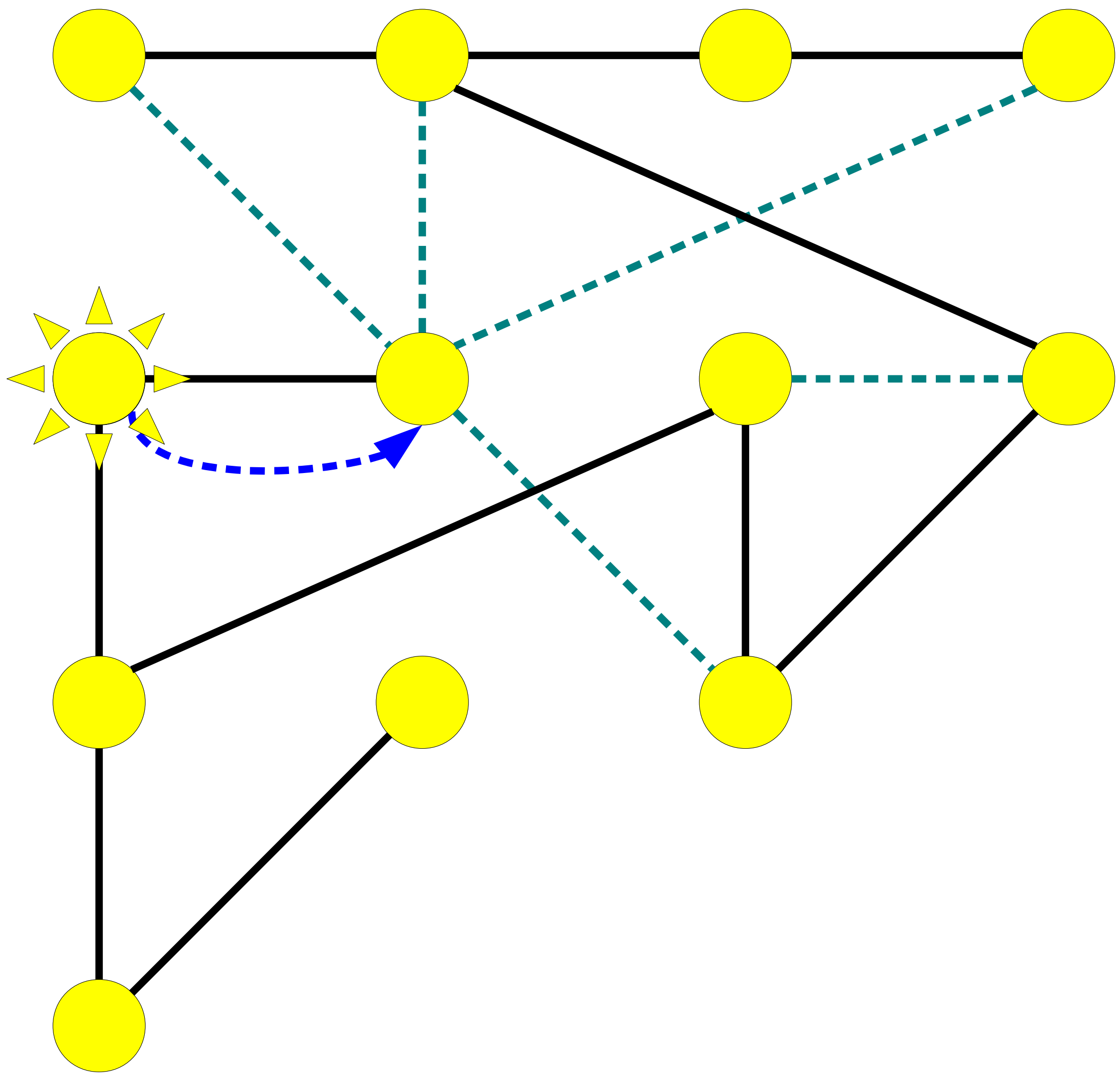
Recursive Depth-First Search



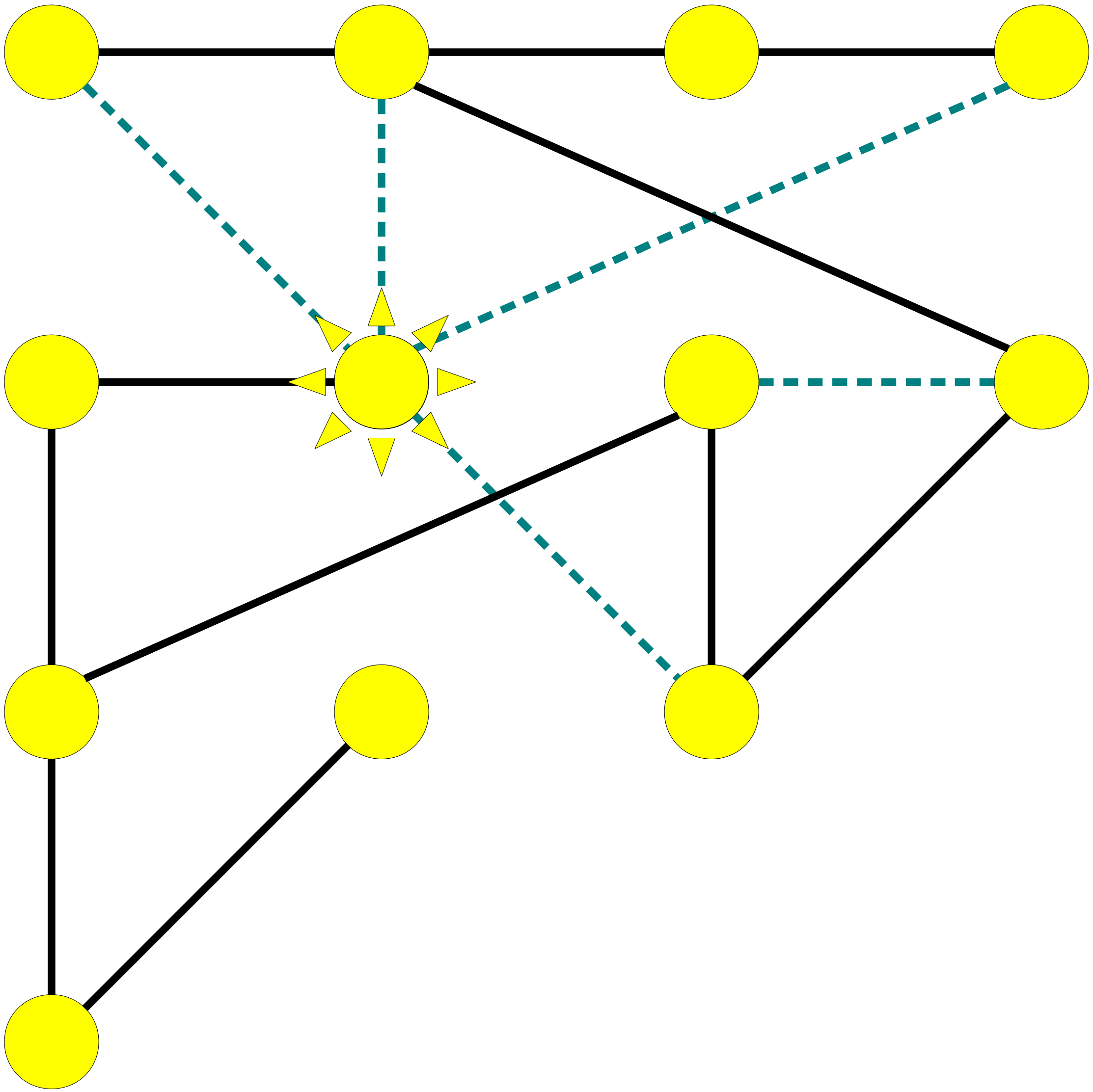
Recursive Depth-First Search



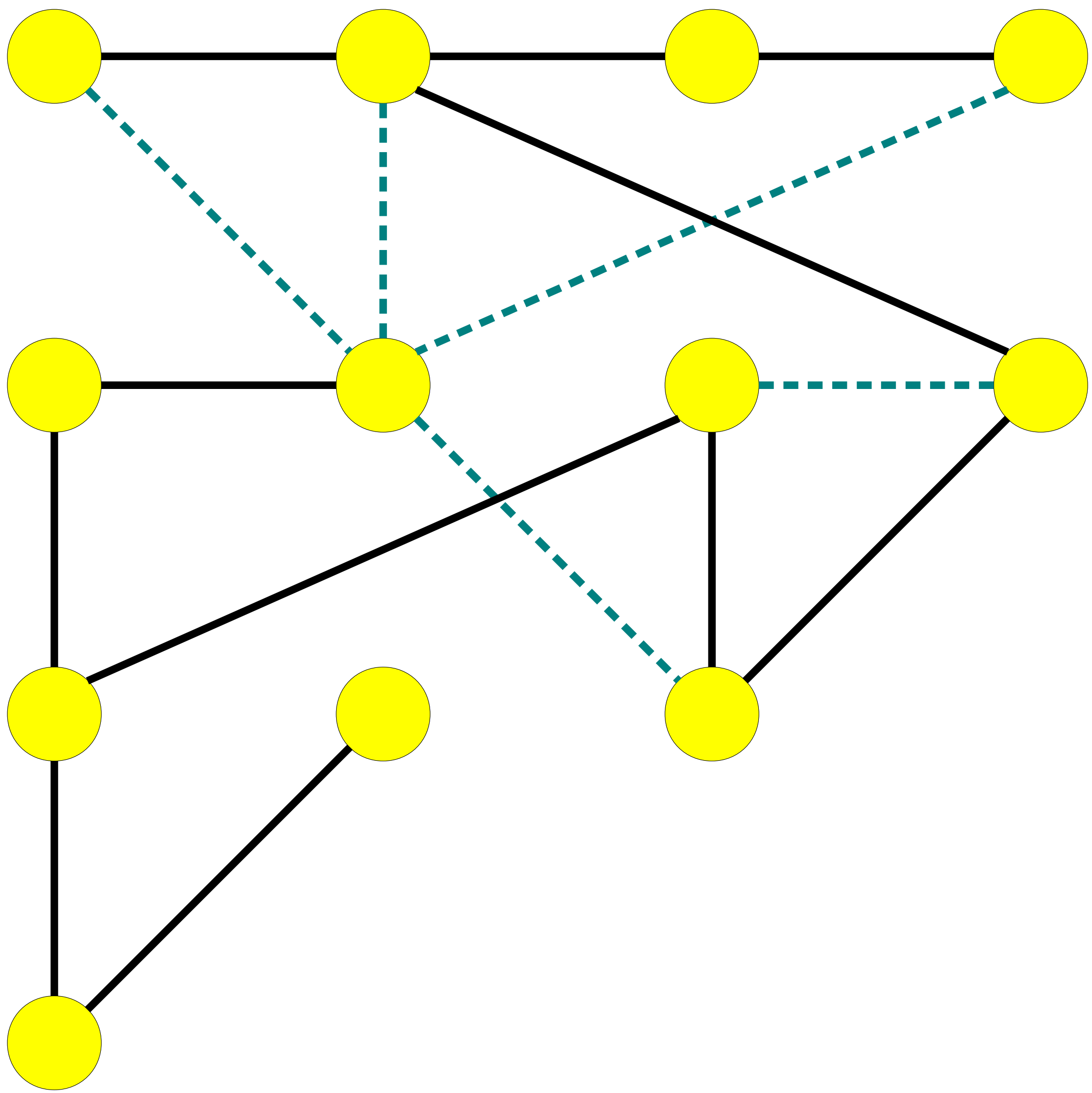
Recursive Depth-First Search



Recursive Depth-First Search



Recursive Depth-First Search



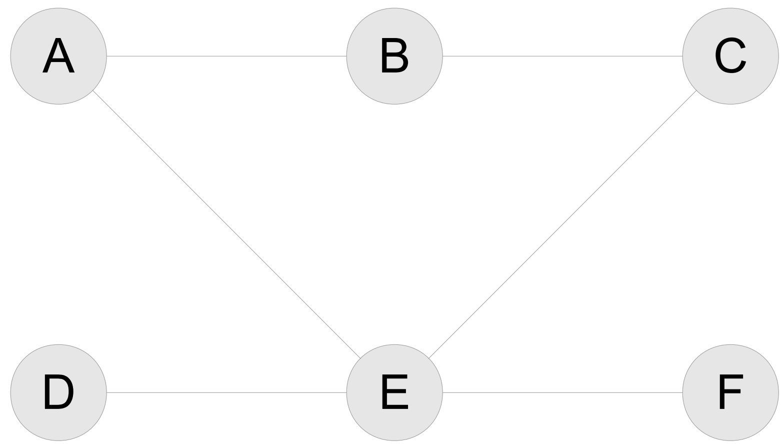
Recursive Depth-First Search

- 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.

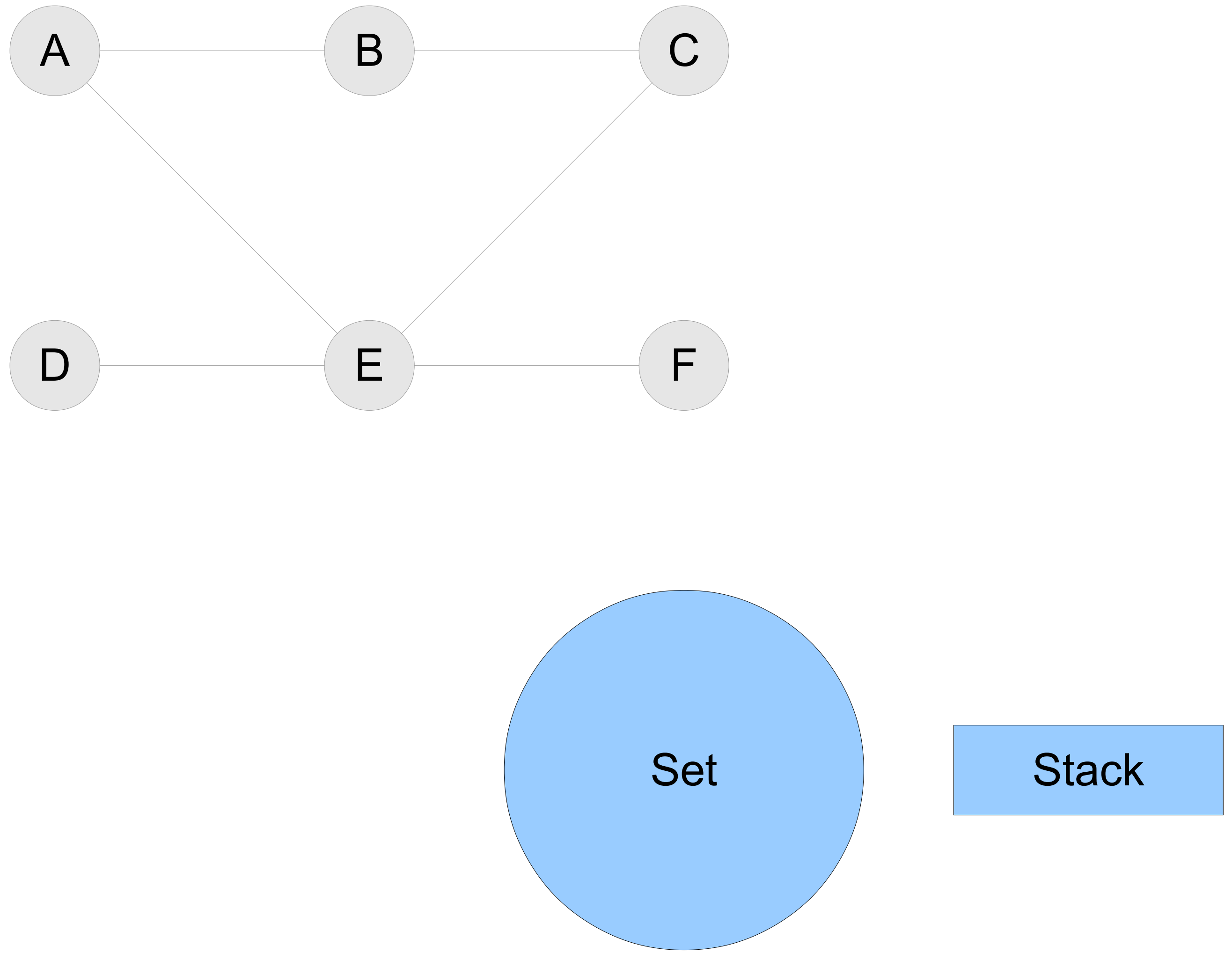
Iterative Depth-First Search

- DFS is most commonly implemented iteratively using a **`Stack`**

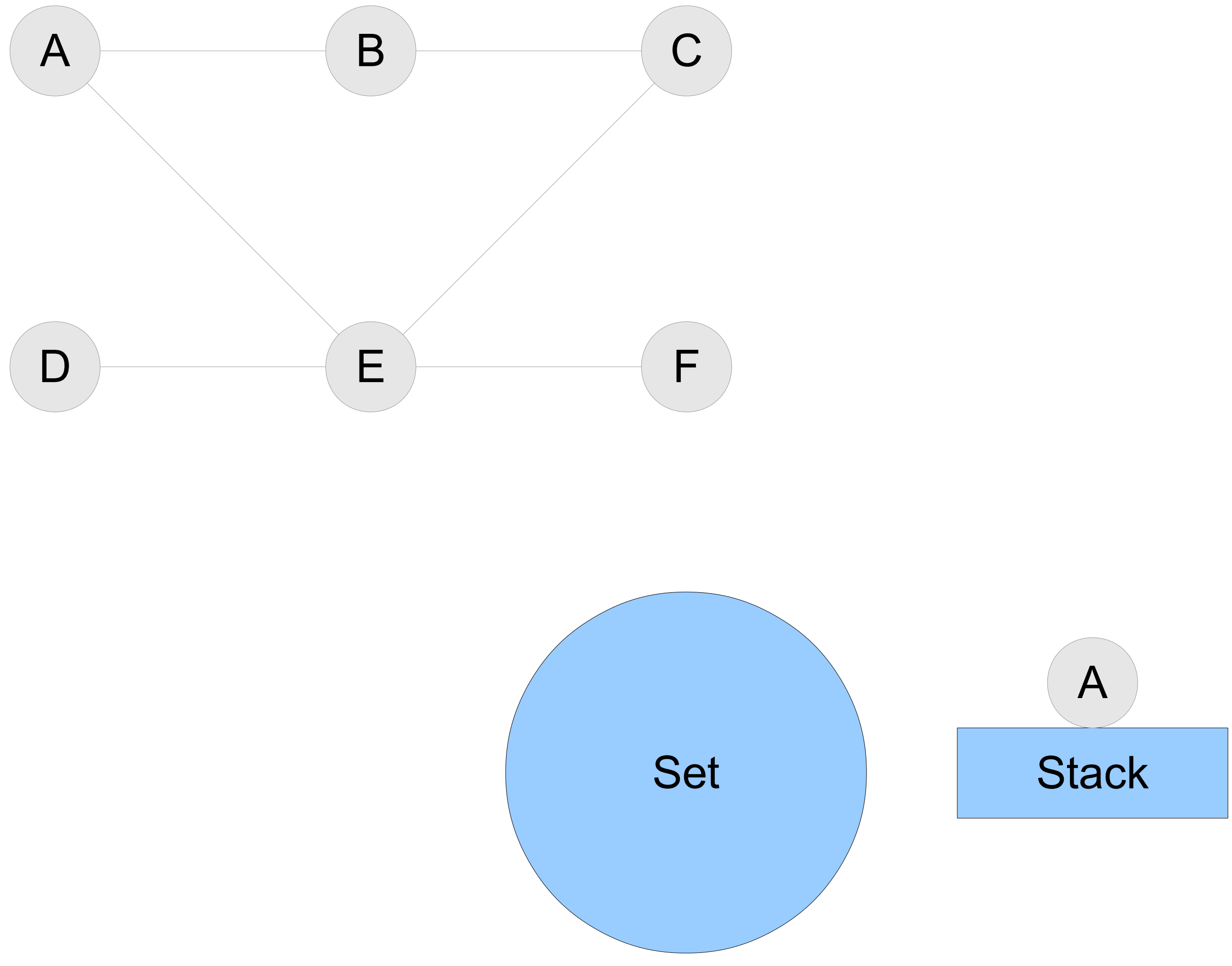
Depth-first search



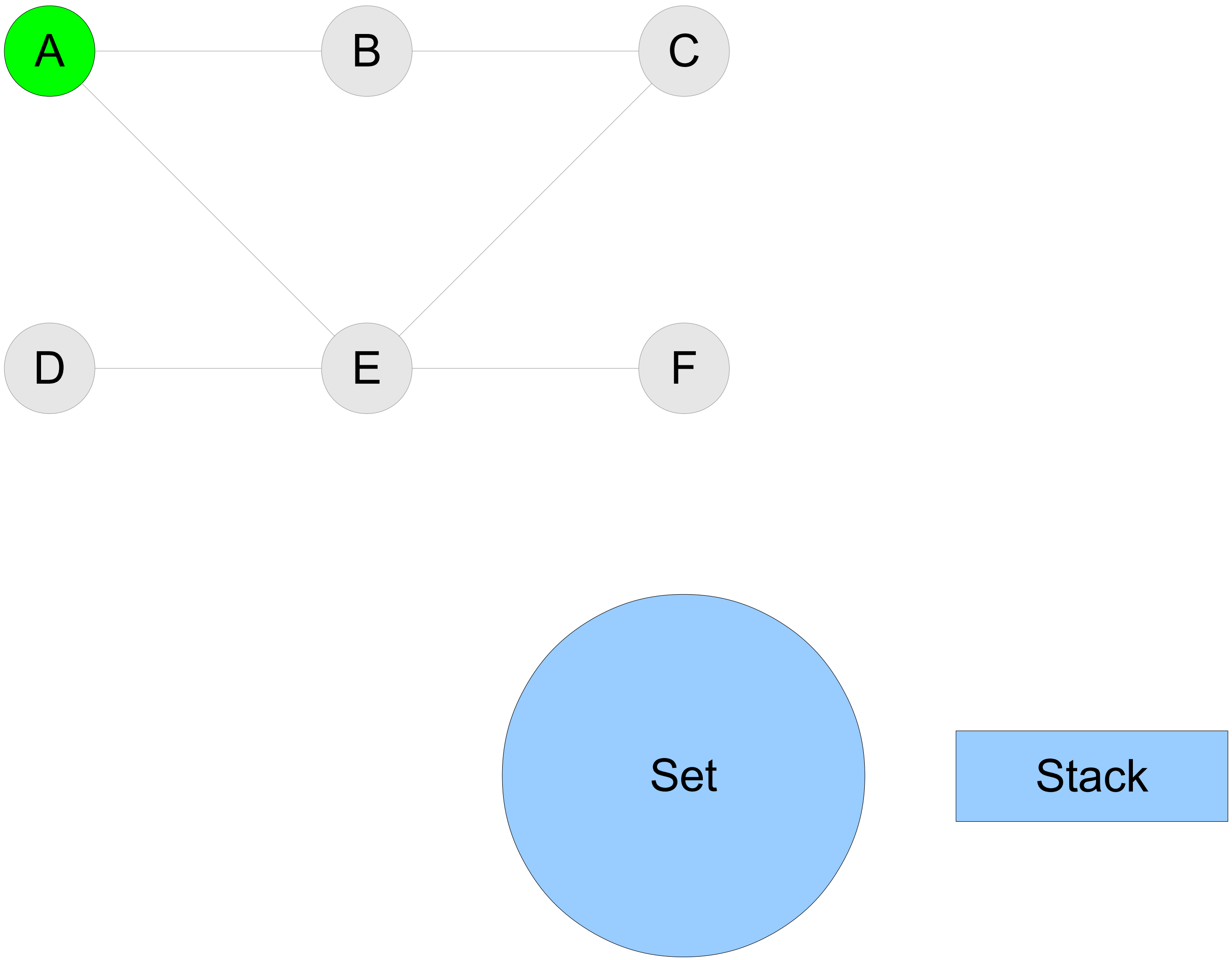
Depth-first search



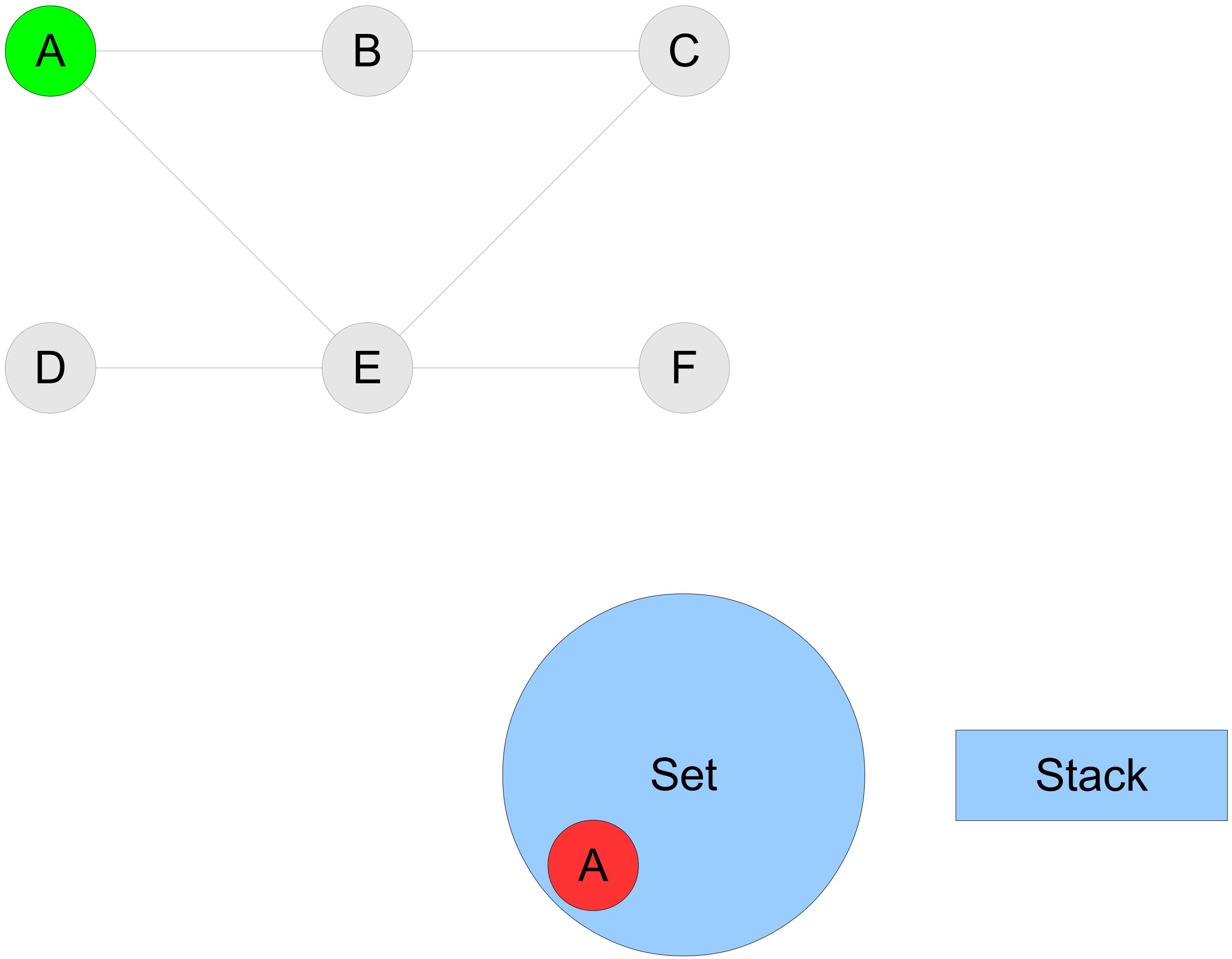
Depth-first search



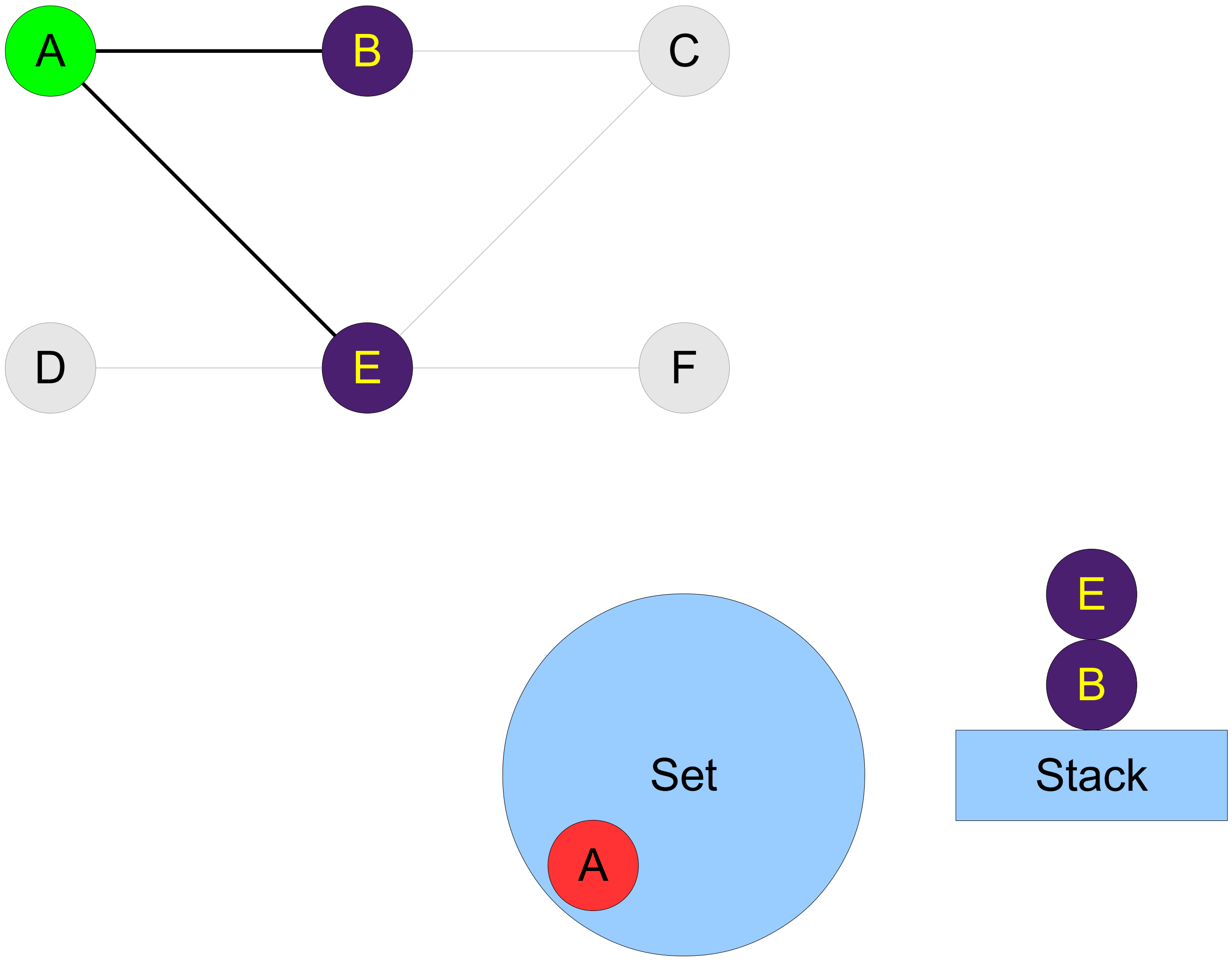
Depth-first search



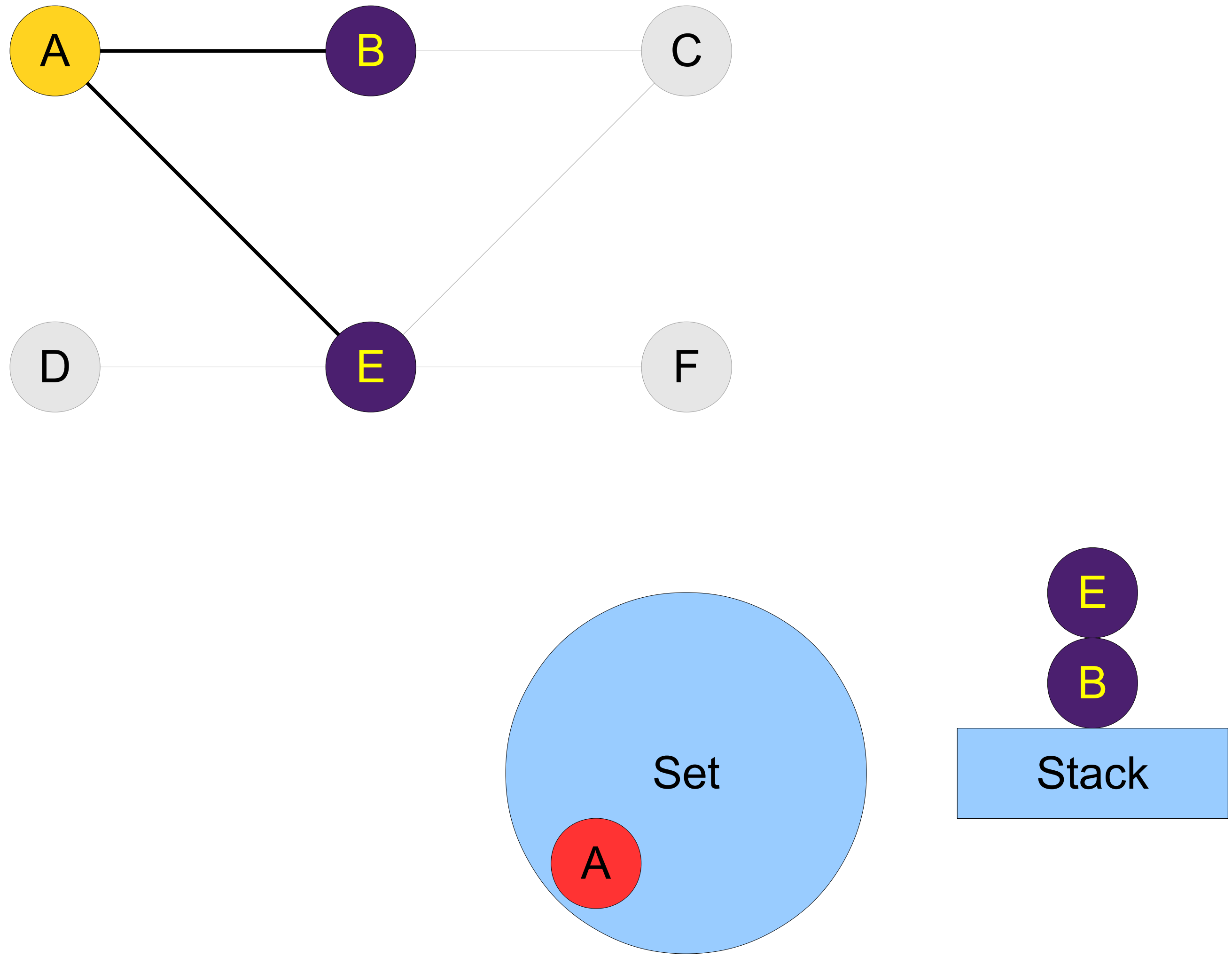
Depth-first search



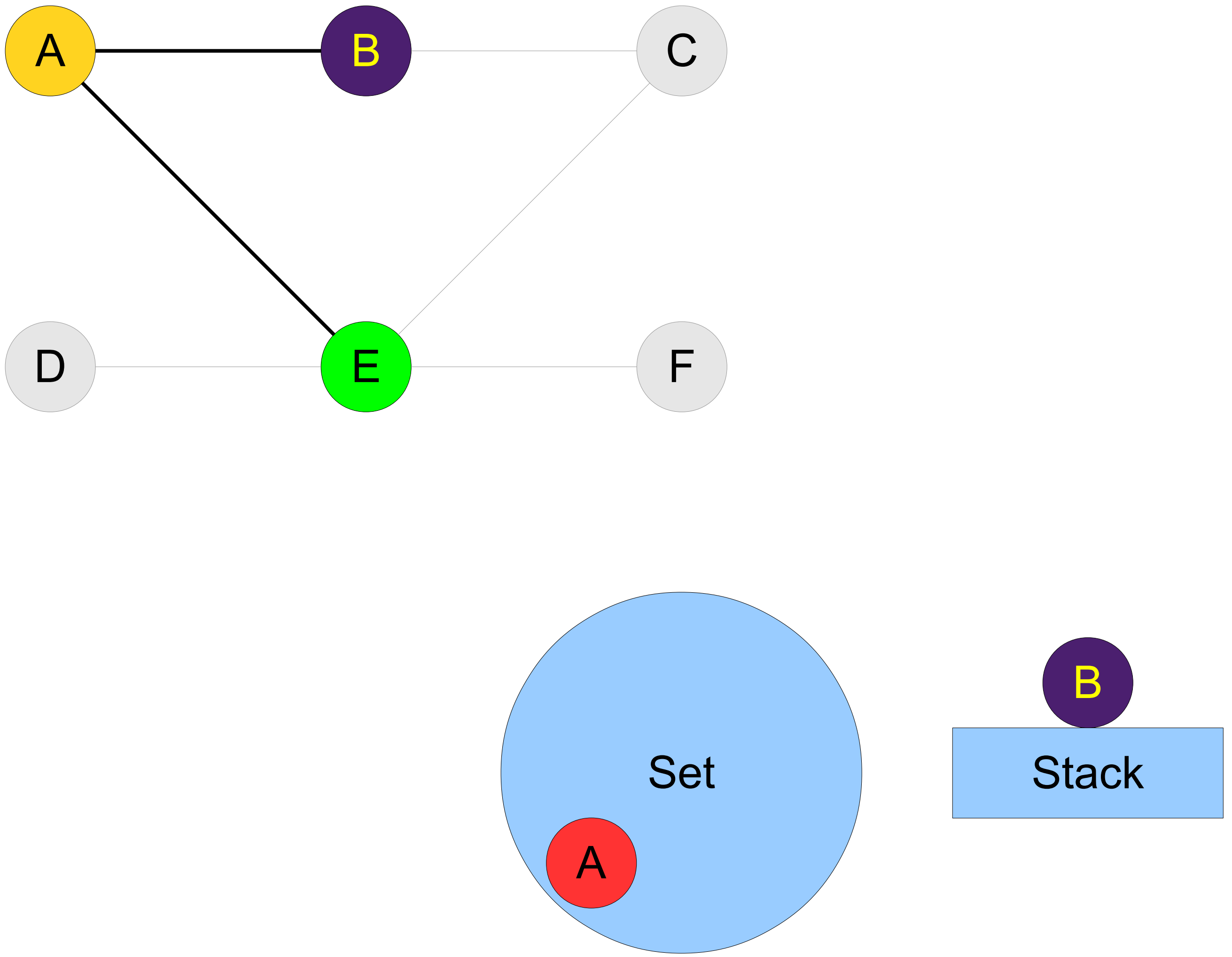
Depth-first search



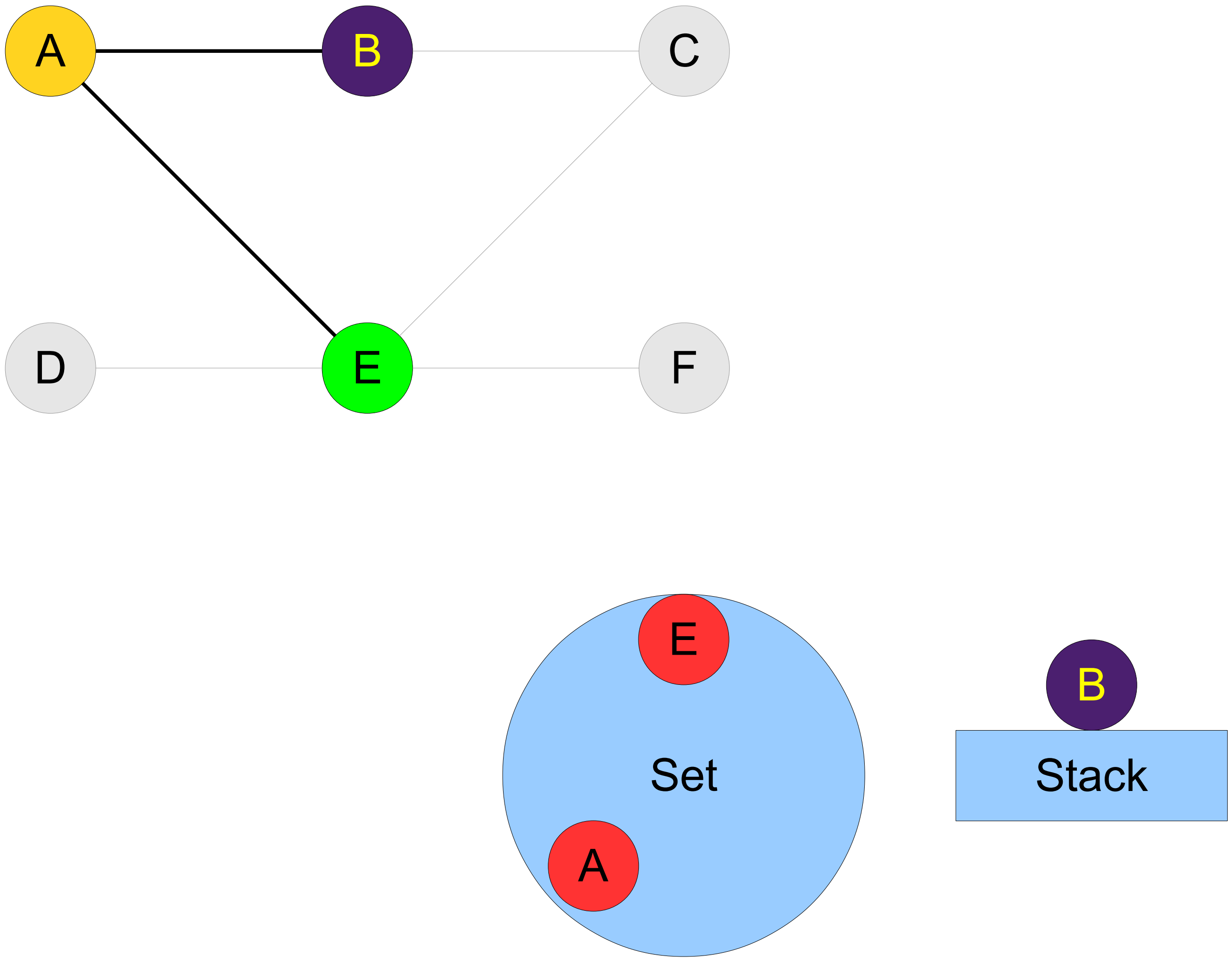
Depth-first search



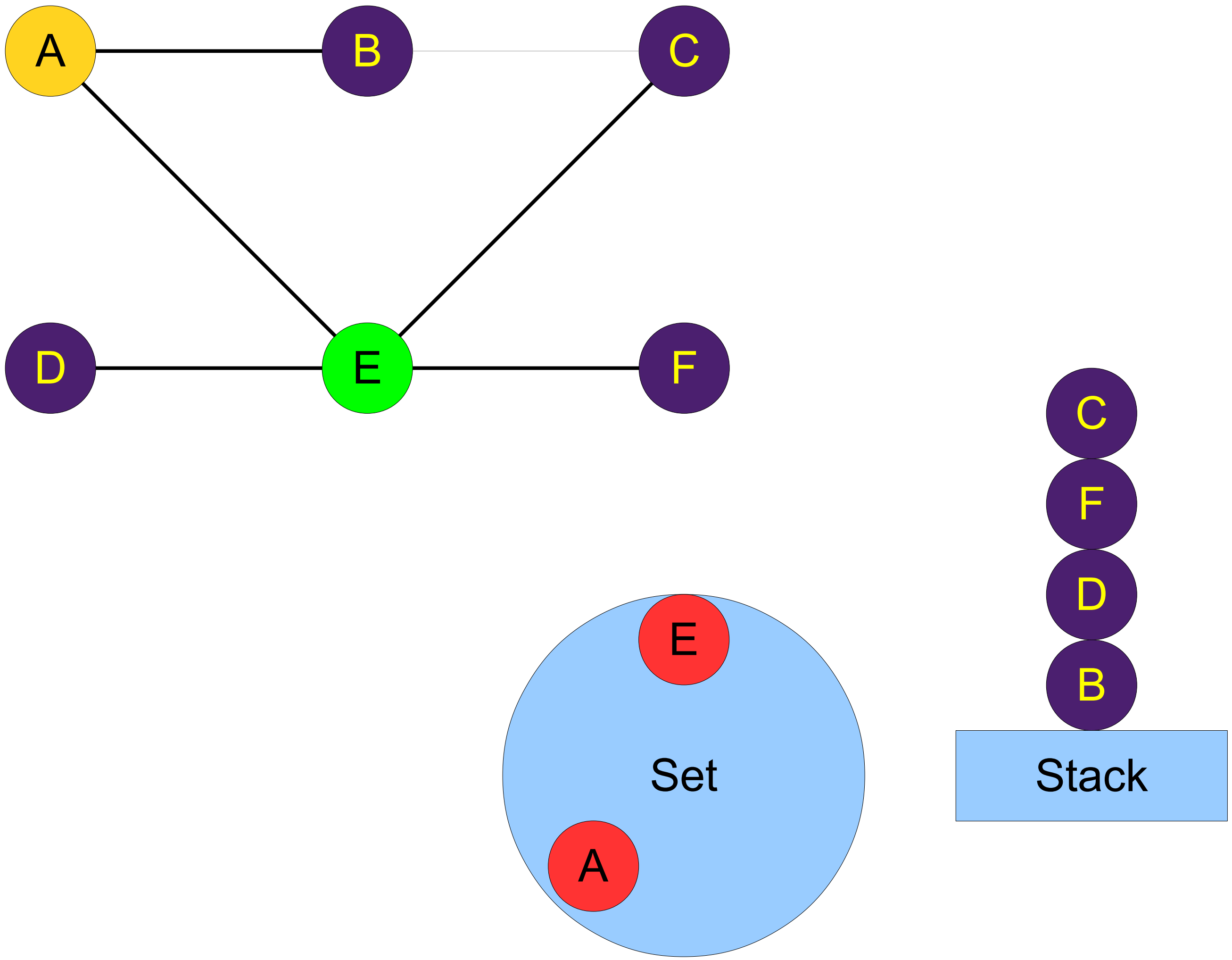
Depth-first search



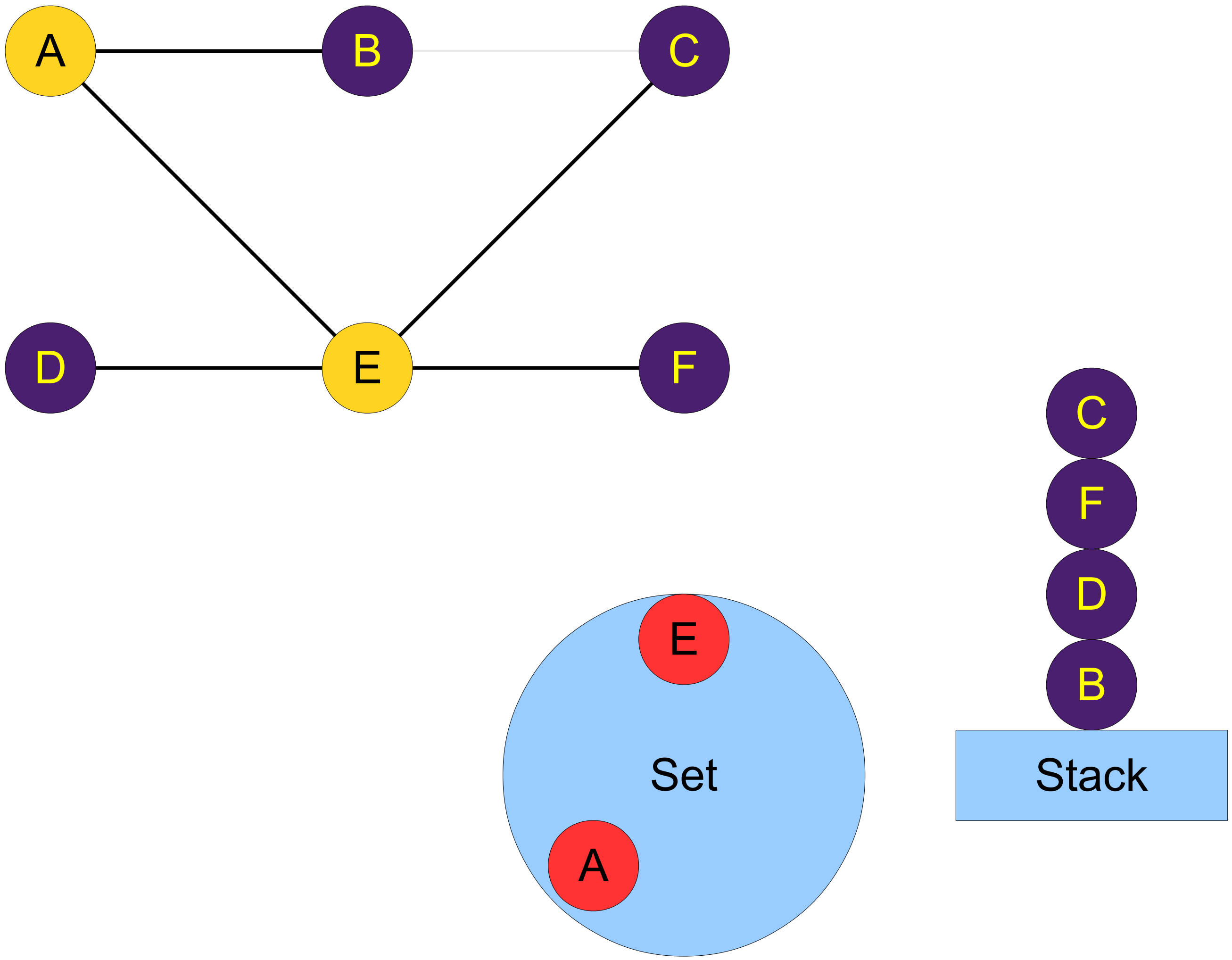
Depth-first search



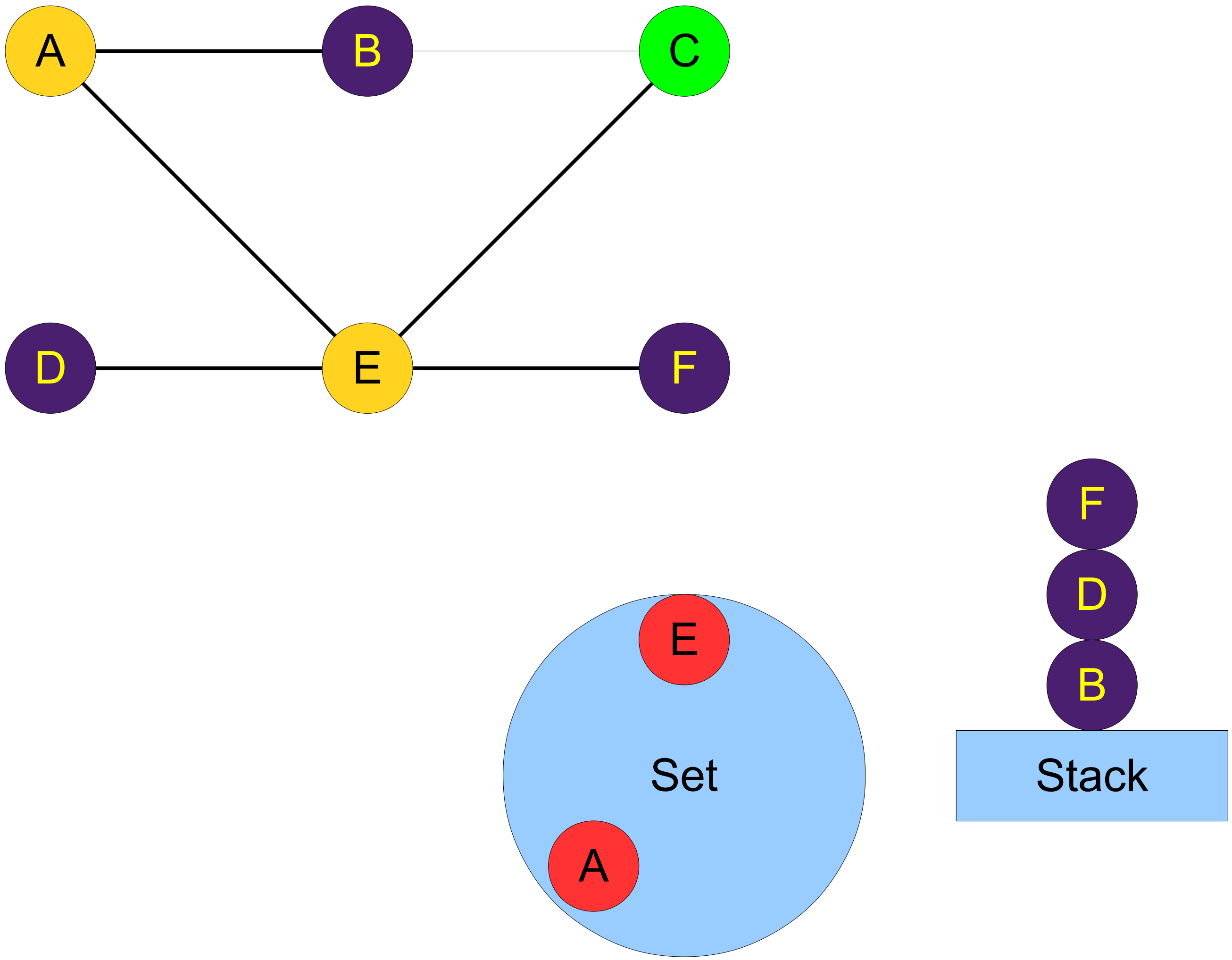
Depth-first search



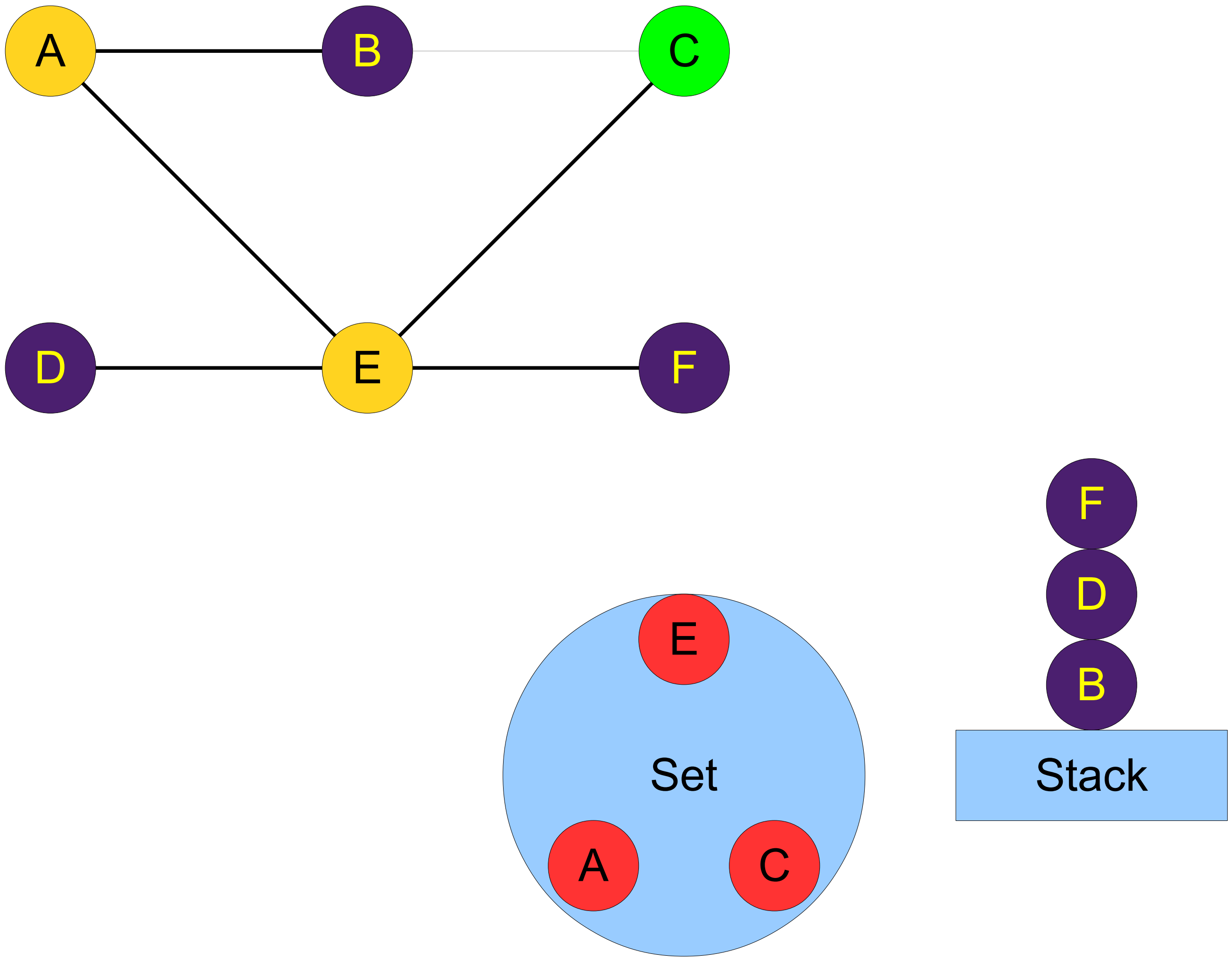
Depth-first search



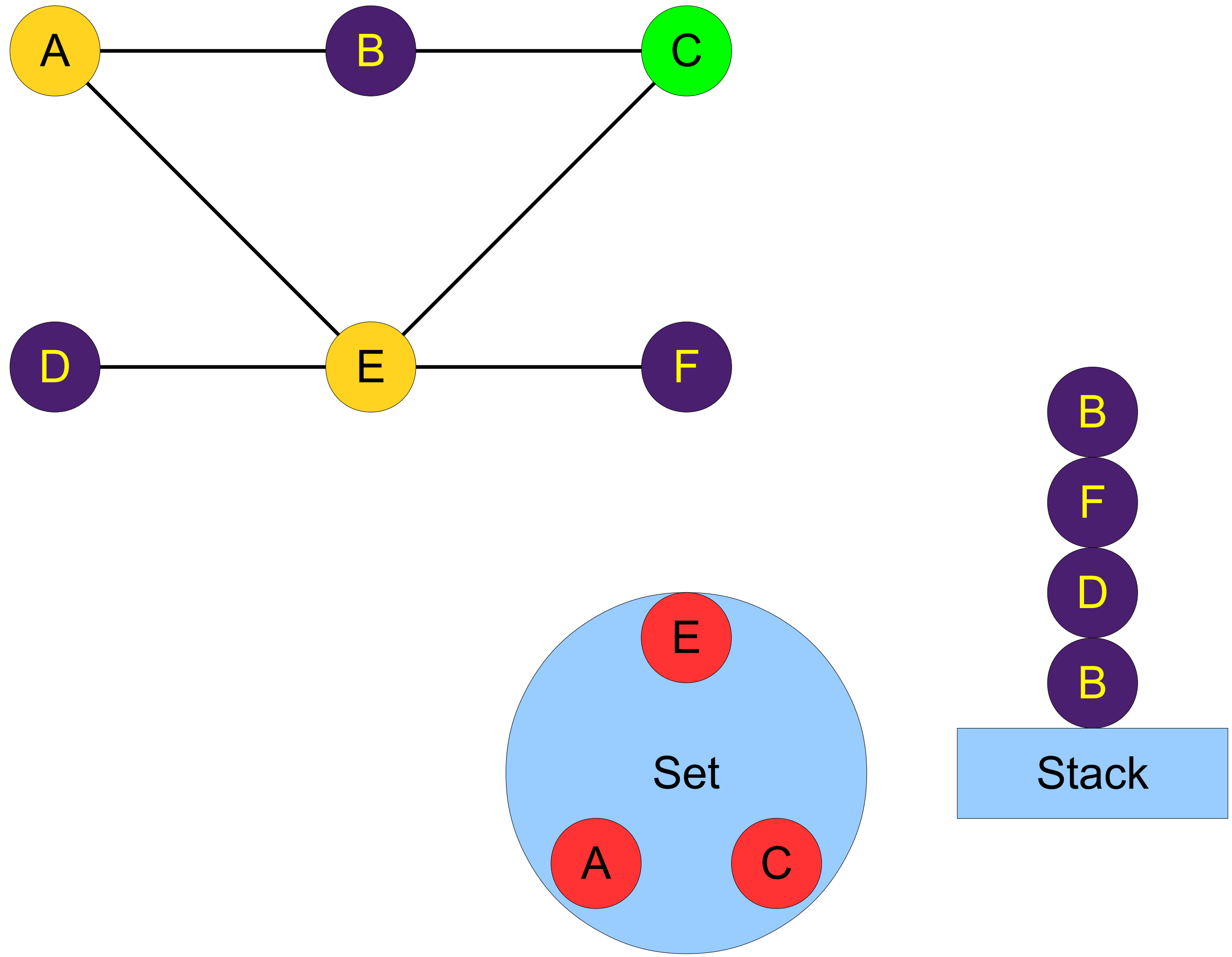
Depth-first search



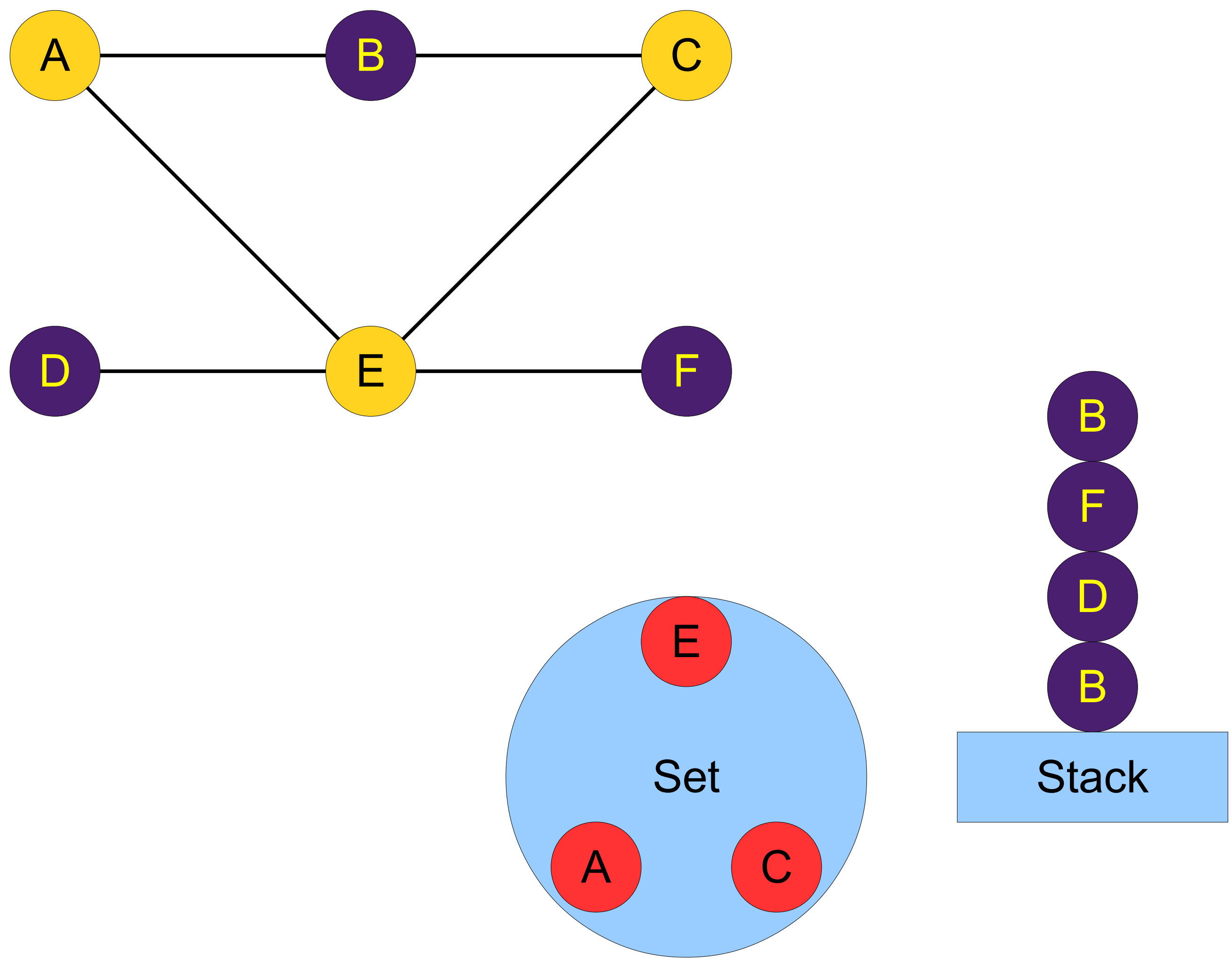
Depth-first search



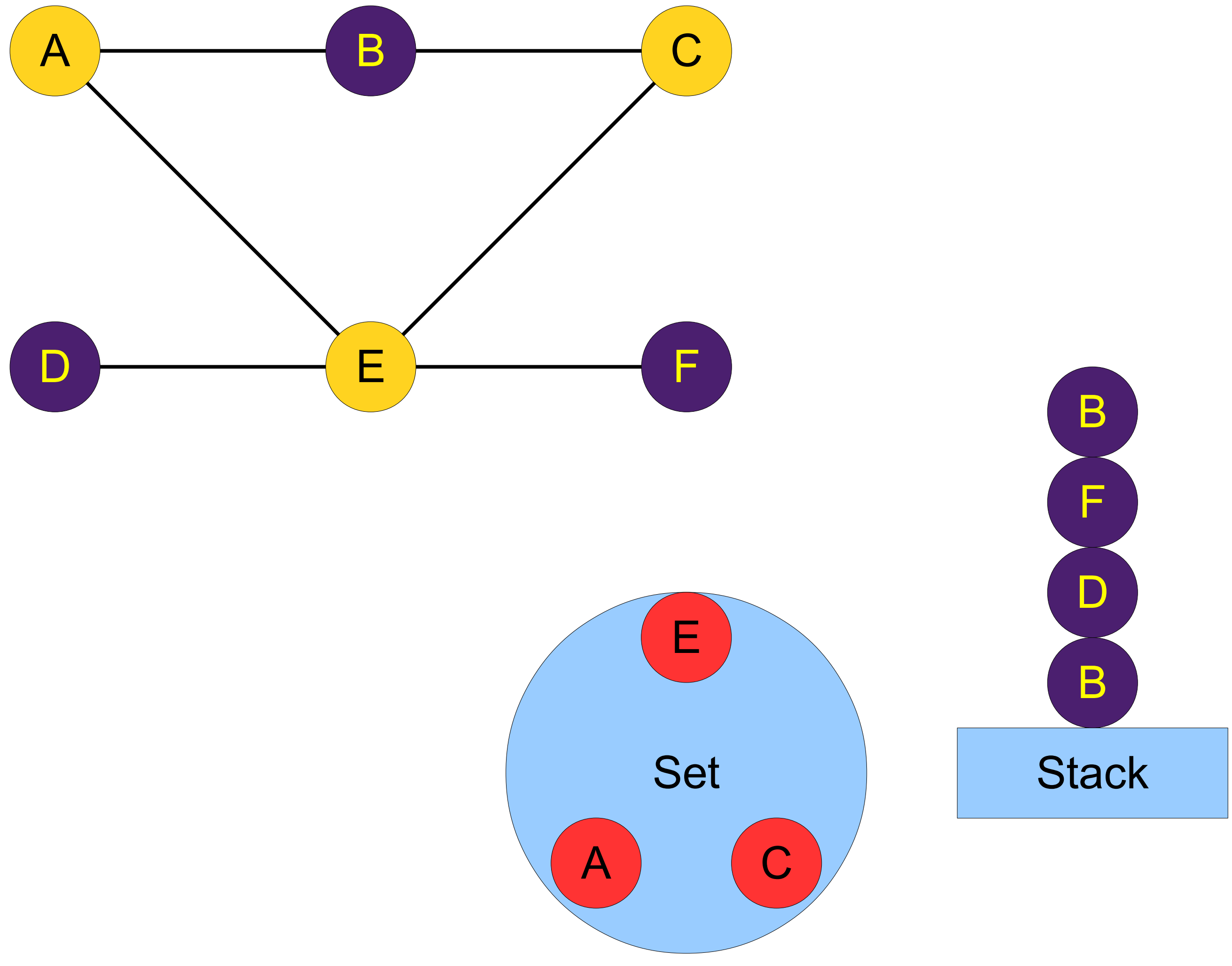
Depth-first search



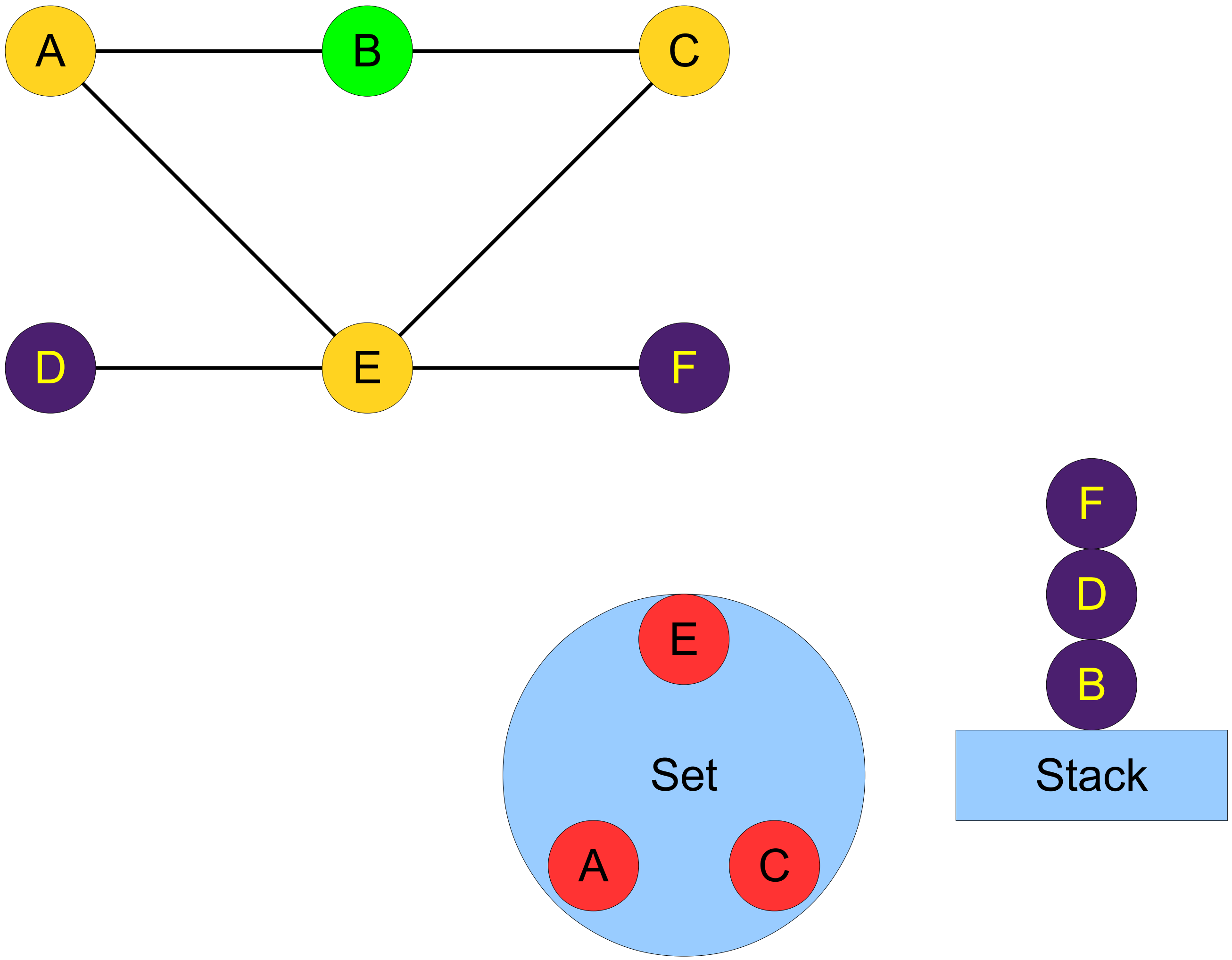
Depth-first search



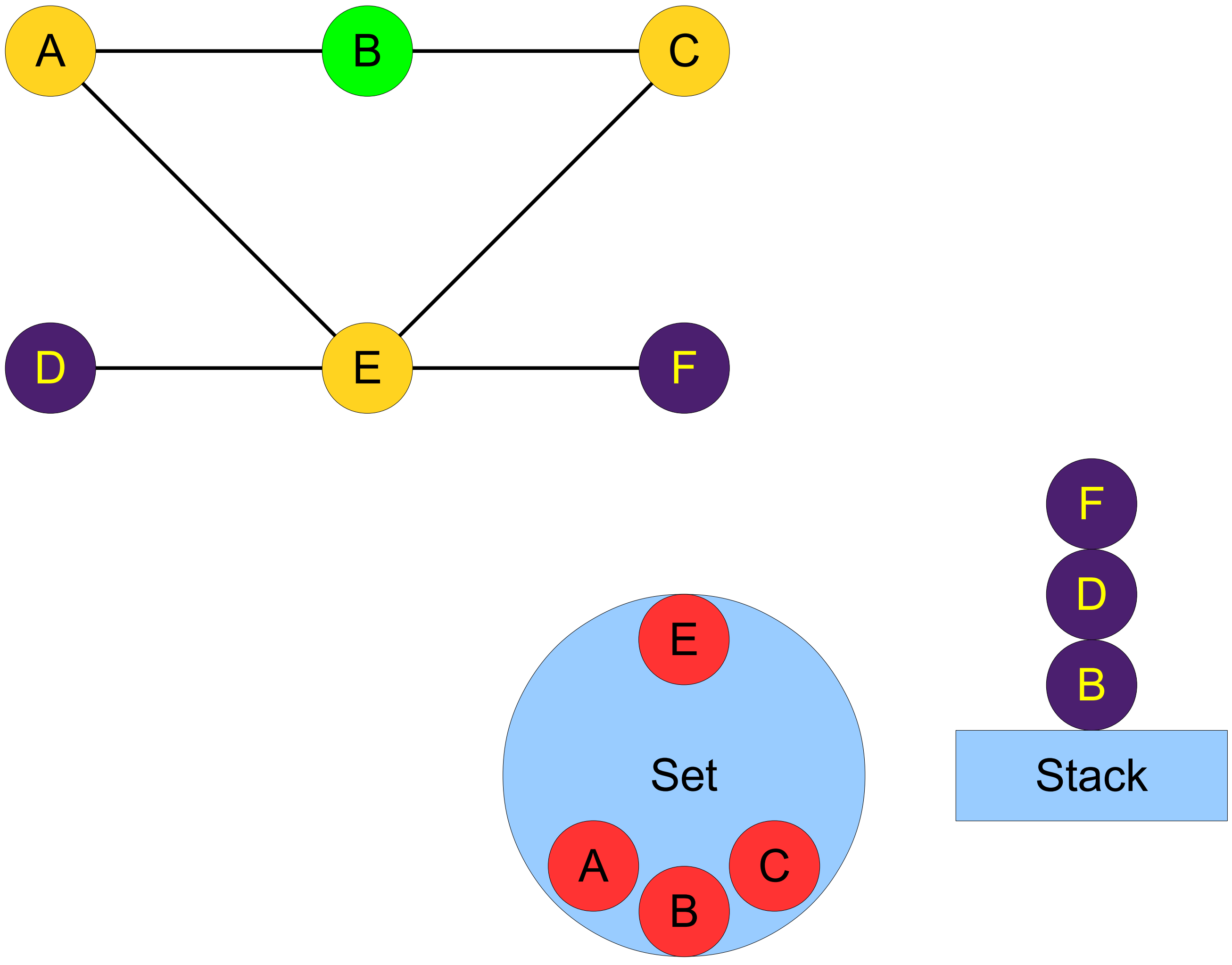
Depth-first search



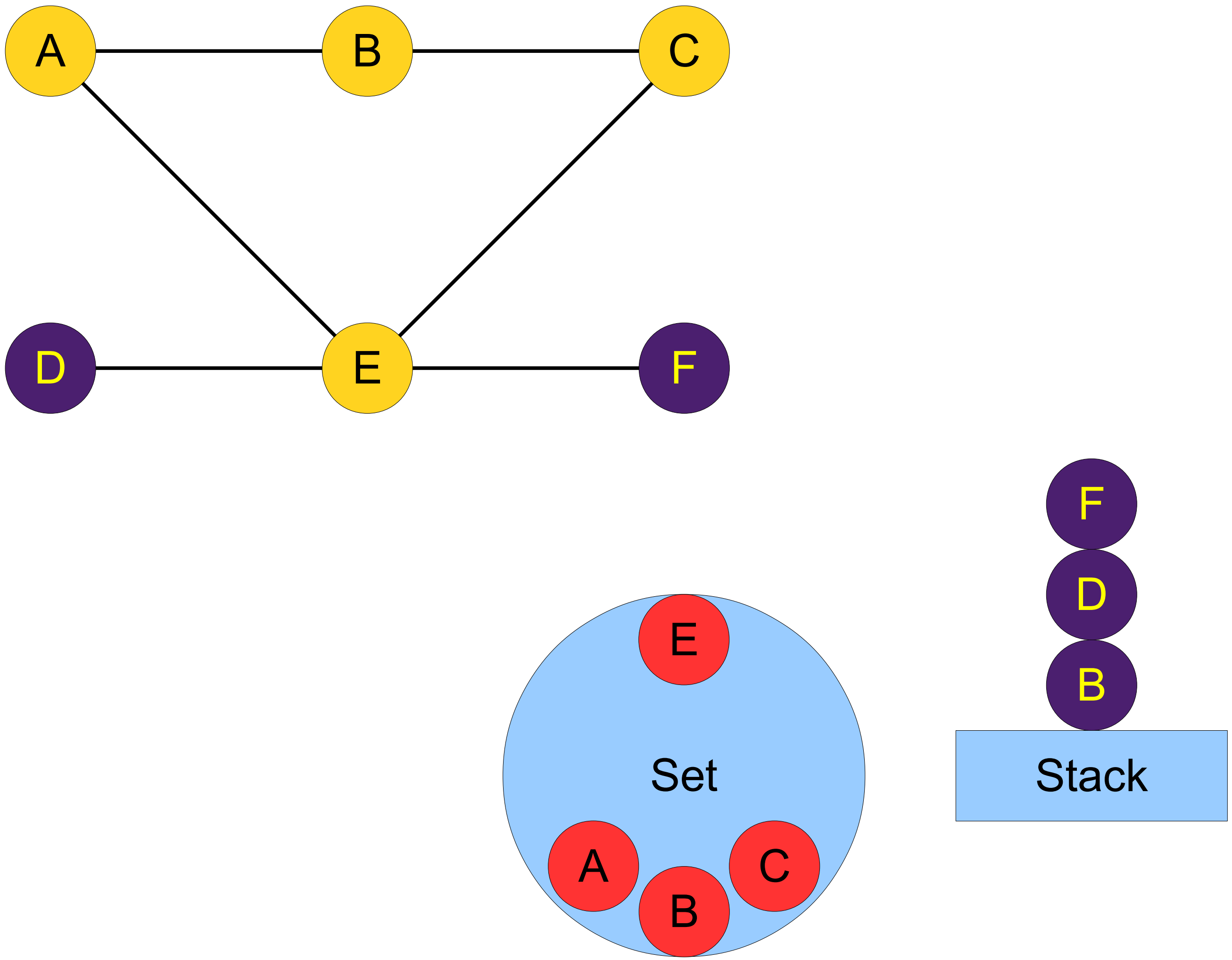
Depth-first search



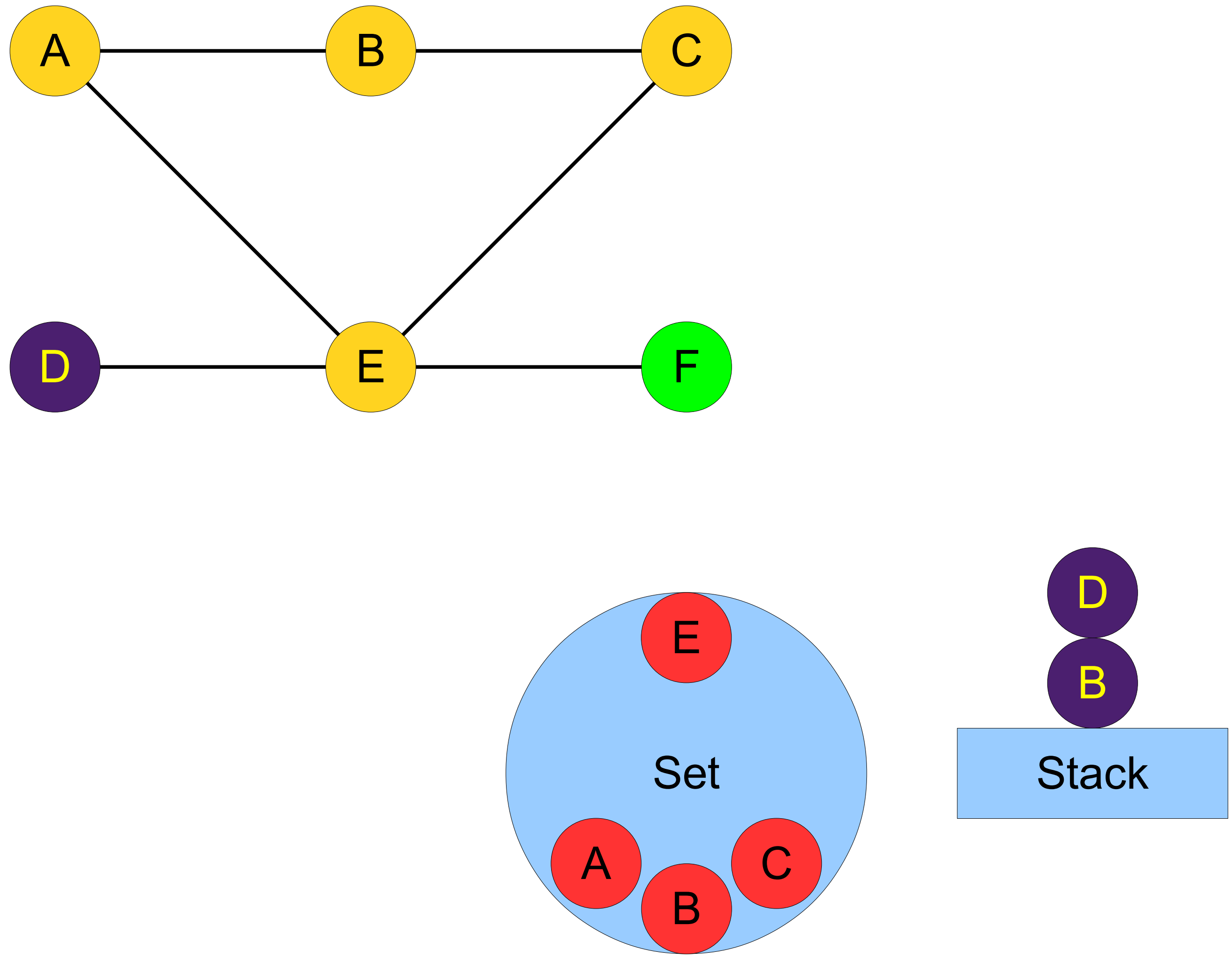
Depth-first search



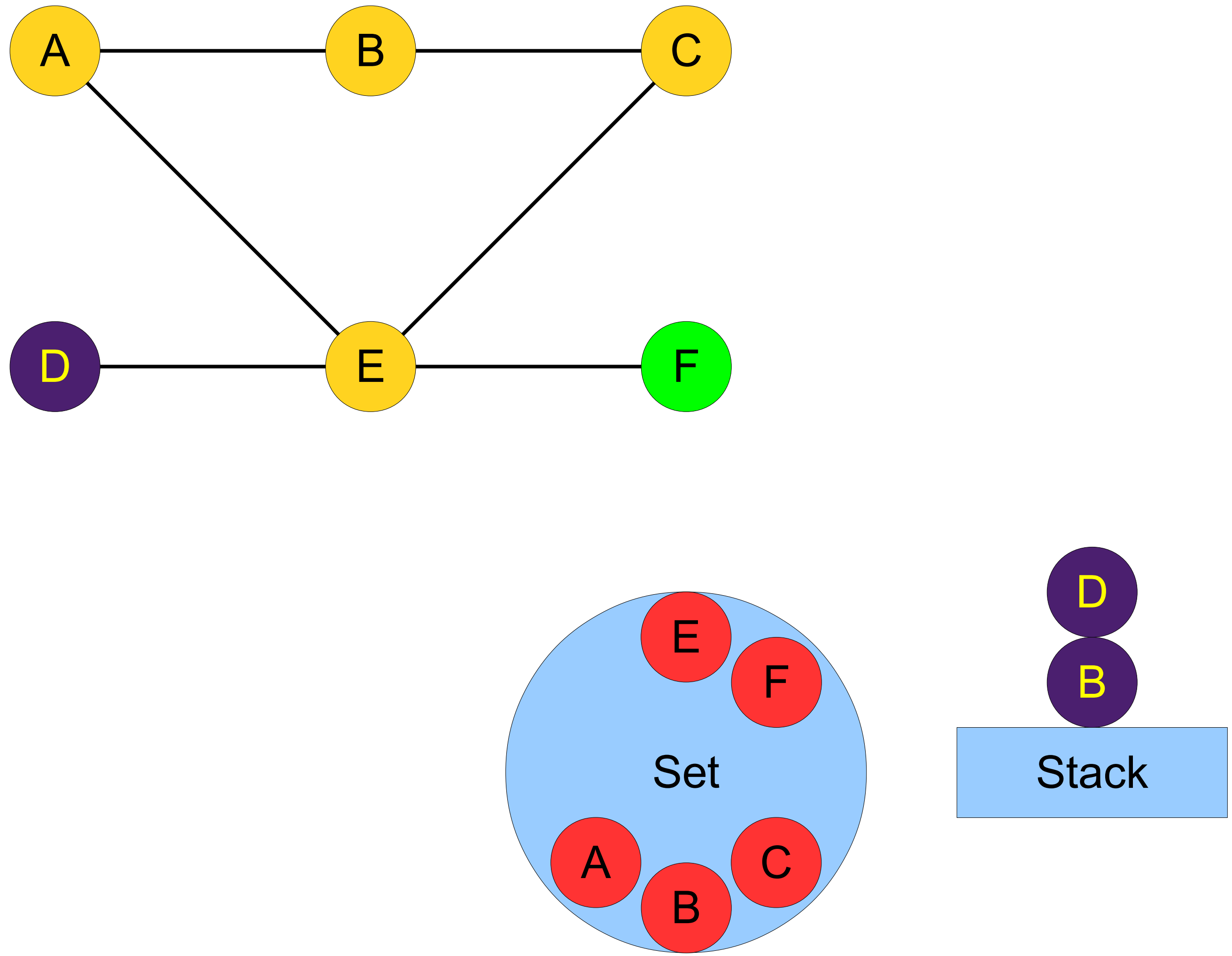
Depth-first search



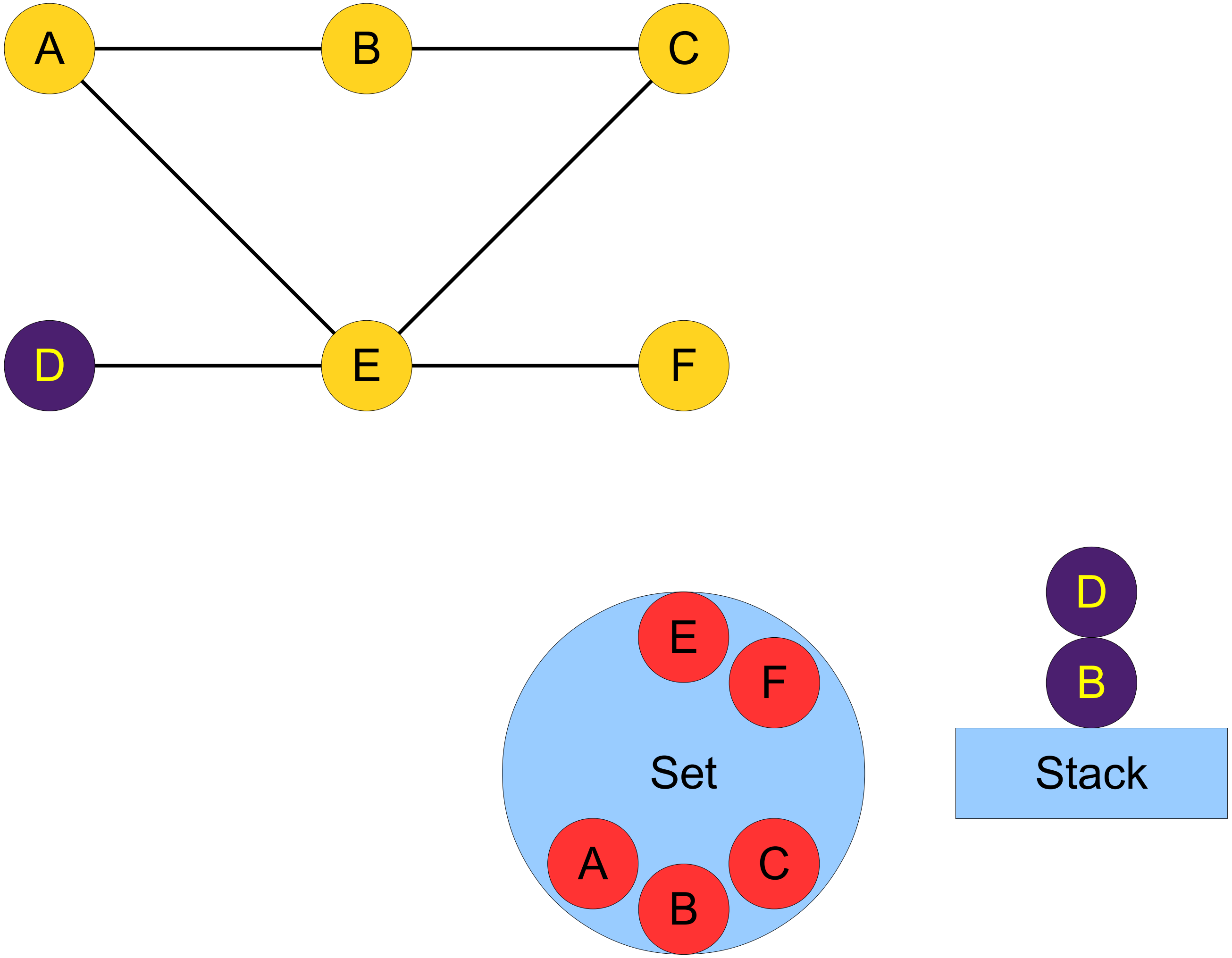
Depth-first search



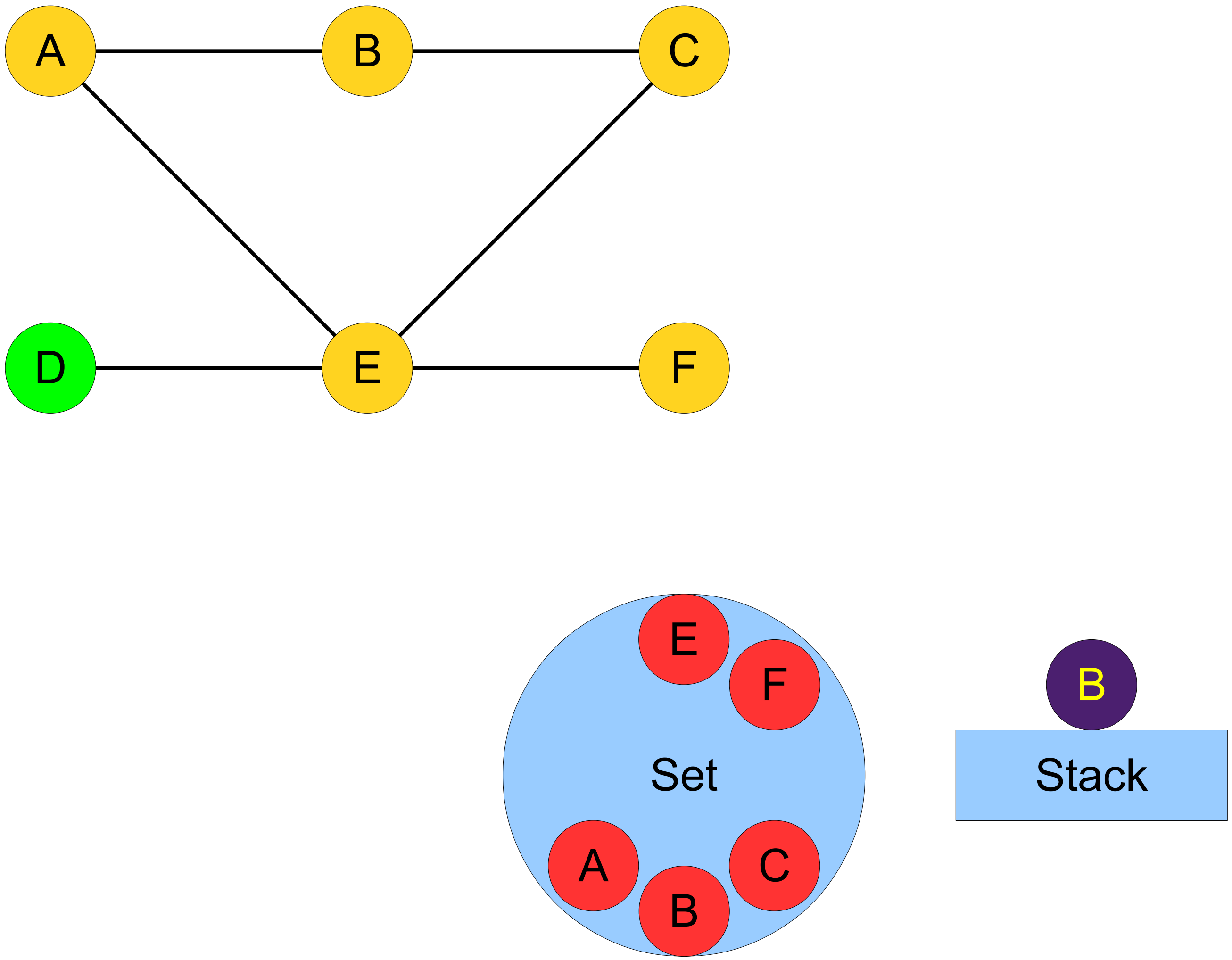
Depth-first search



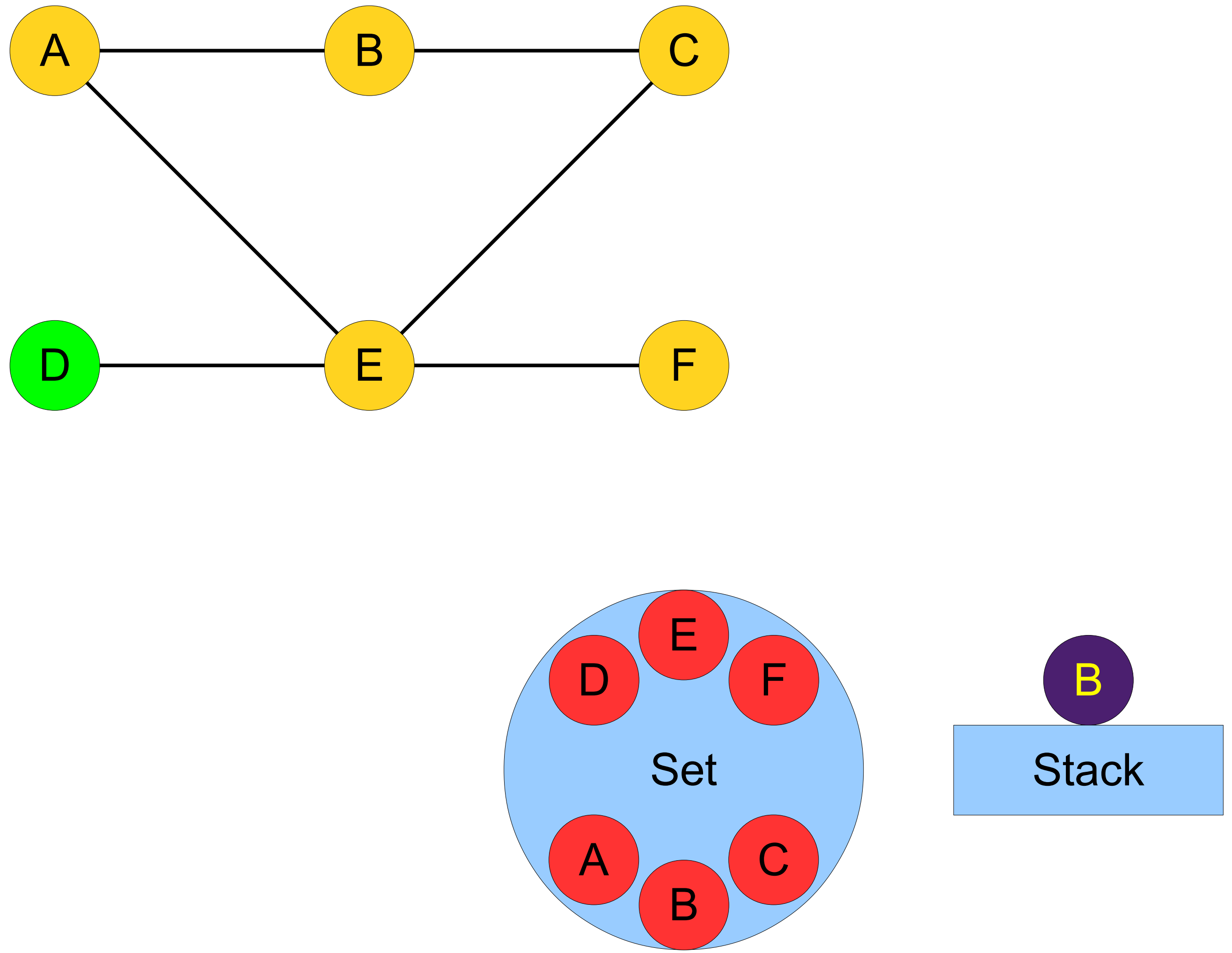
Depth-first search



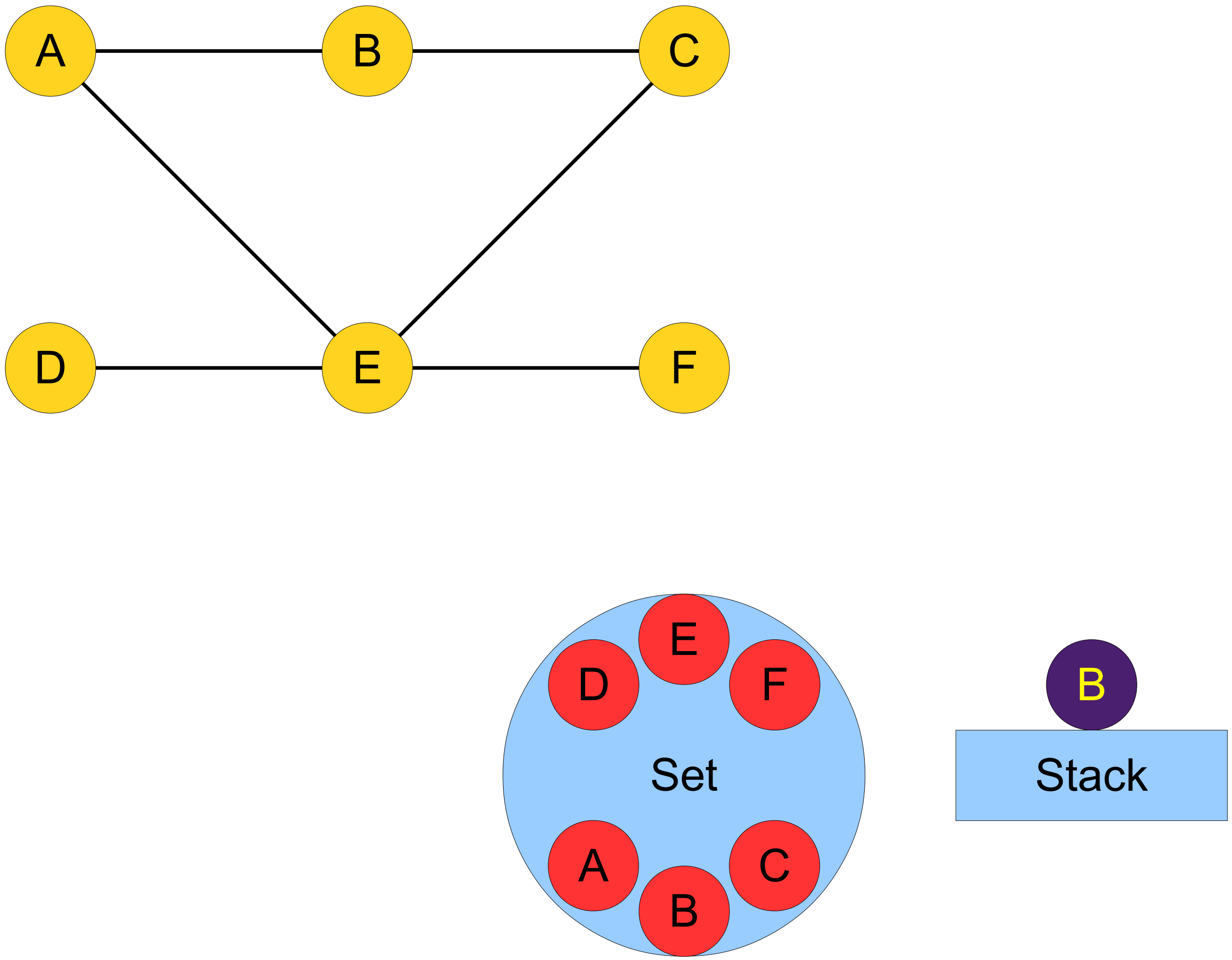
Depth-first search



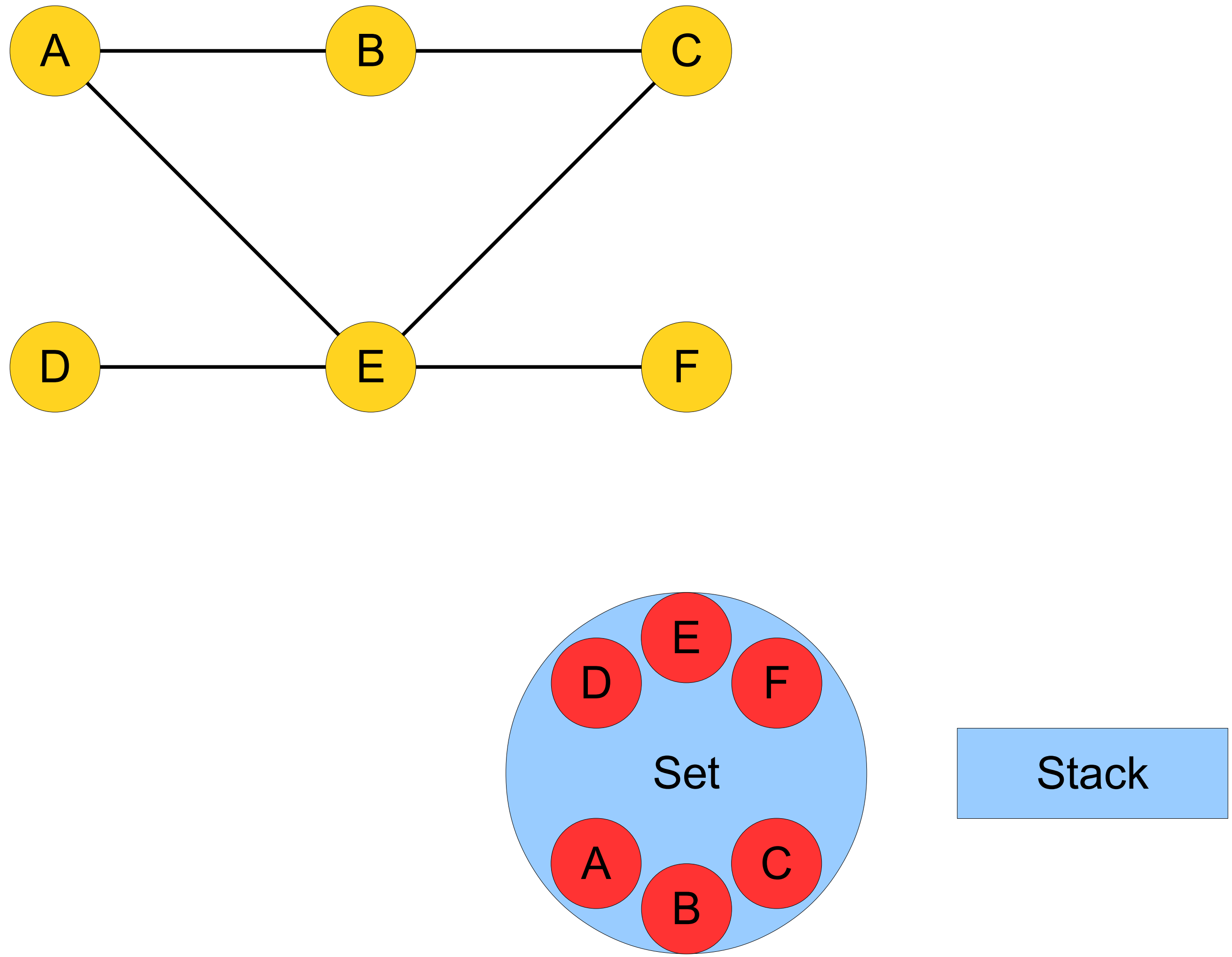
Depth-first search



Depth-first search



Depth-first search



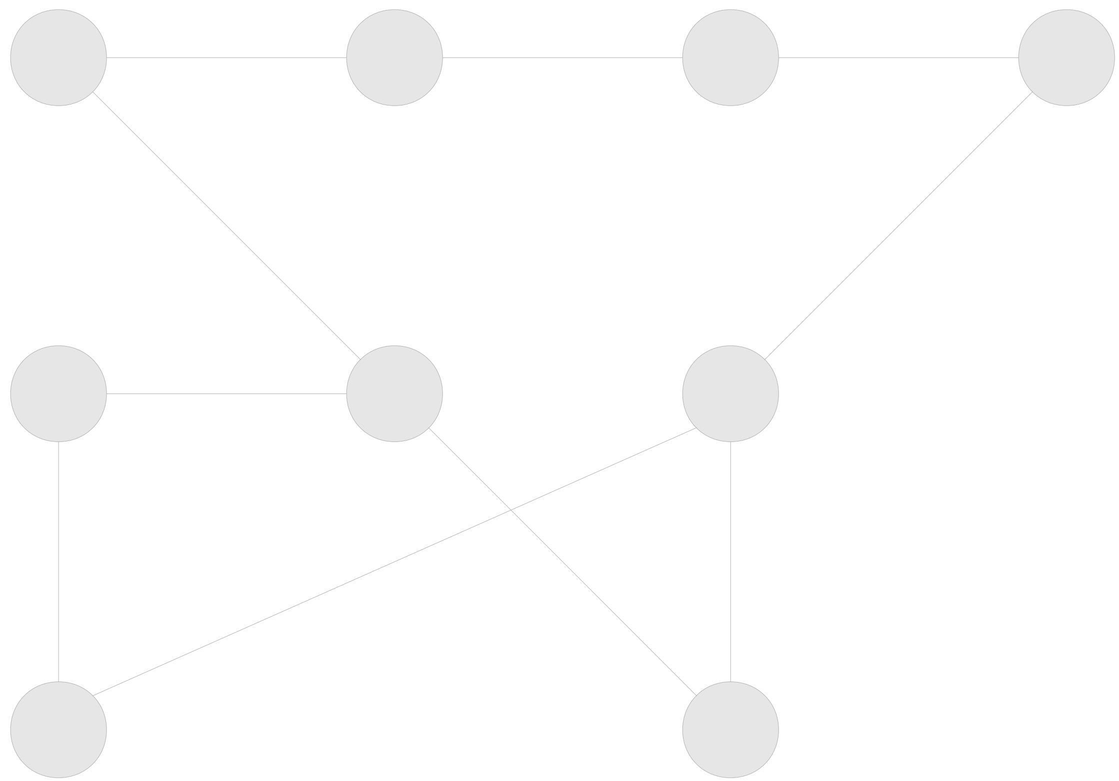
Coding Depth-First Search

DFS and Connected Components

- Detecting connected components becomes relatively straightforward once we have Depth First Search.
- Not going to code it up, but I encourage you to!

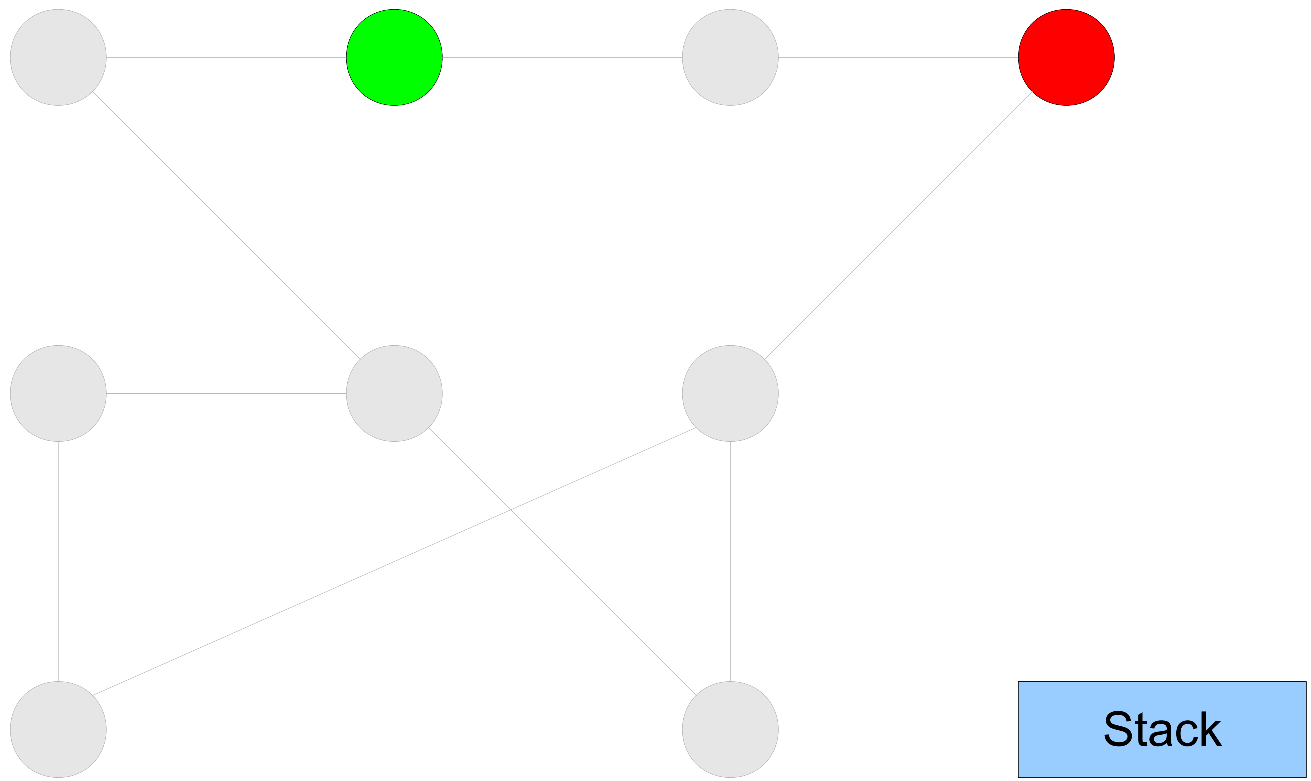
Problems with DFS

- Useful when trying to explore everything.
- Not good at finding shortest paths.



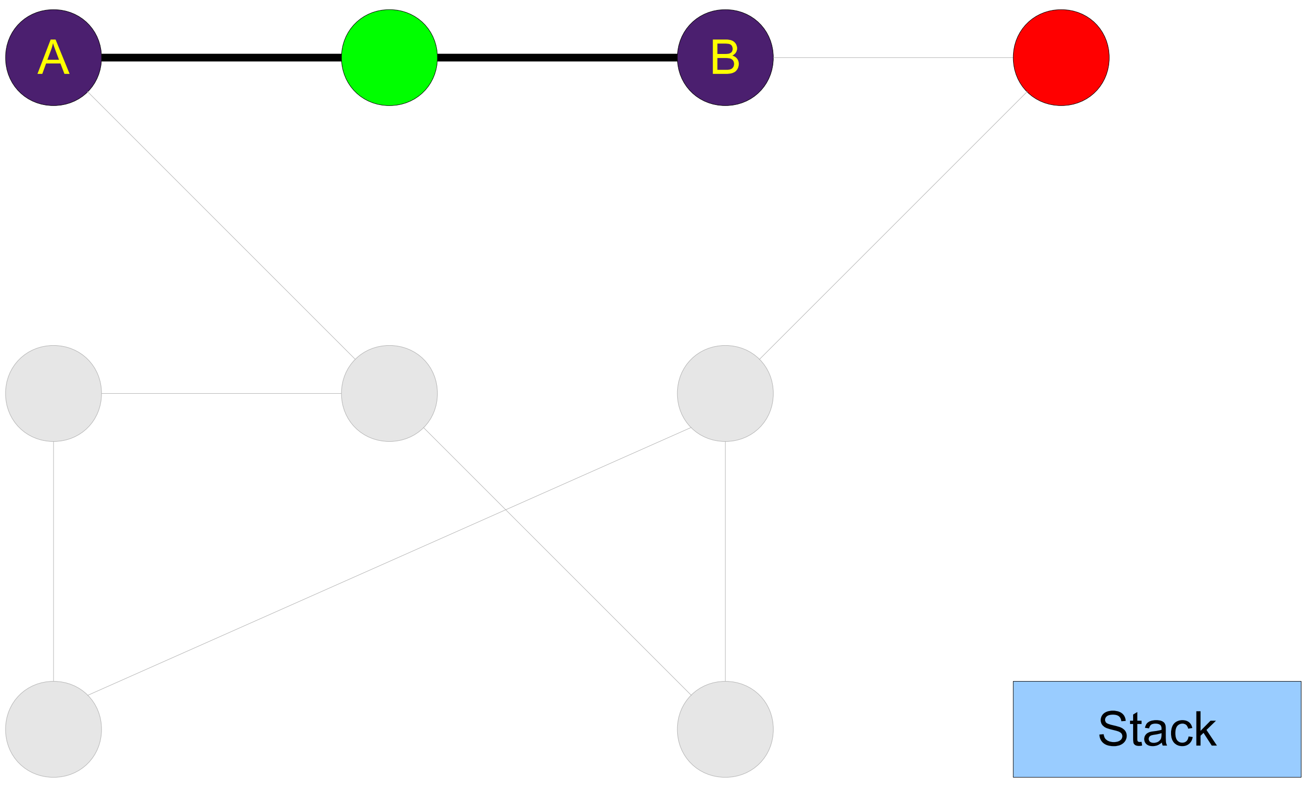
Problems with DFS

- Useful when trying to explore everything.
- Not good at finding shortest paths.



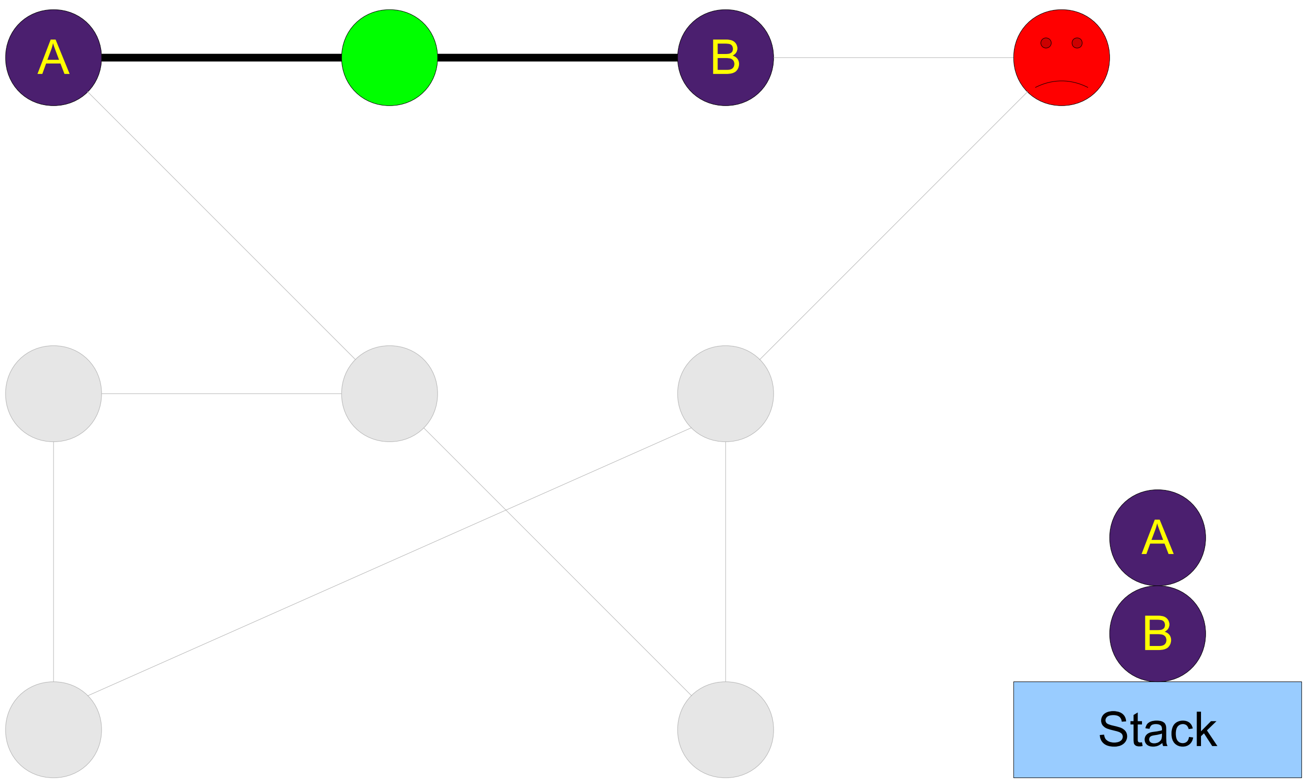
Problems with DFS

- Useful when trying to explore everything.
- Not good at finding shortest paths.



Problems with DFS

- Useful when trying to explore everything.
- Not good at finding shortest paths.



Breadth-First Search

STANFORD UNIVERSITY

PARKING AND CIRCULATION MAP 2010-11

- **Parking & Transportation Services** *Parking permit sales, 7:30am - 5pm (M-F)*
- **Visitor Parking** *Pay meter or time limit, 8am - 4pm (M-F)*
- **Visitor Parking / Pay Station** *Pay at machine, 8am - 4pm (M-F)* *Pay by credit card (Visa/Mastercard) or cash*
- **Event Parking** *6am - 4pm (M-F)*
- **Bus Parking** *6am - 4pm (M-F)*
- **Motorcycle Parking** *6am - 4pm (M-F)*

Commuter Parking *6am - 4pm (M-F)*

- A Permit
- C Permit

Resident Parking *24 hours, 7 days*

- EA Permit
- SJ Permit
- ES Permit
- SO Permit
- SH Permit
- WE Permit

Many lots are signed only at the entrance. In "shared" lots (e.g., those posted "EA or C") you may park with either permit designated.

Disability Parking

Any State issued disability parking permit is valid in any parking space on campus. There are DP spaces at most buildings. For more info, see *maps.stanford.edu/ada*

- **Parking Lot Number**
- **Parking Structure Number**
- **Underground Parking Structure Number**
- **Visitor Center** *650.723.2560*
- **Car Rental**
- **Car-Sharing Vehicle**
- **Athletic Fields**
- **Construction**
- **Pedestrian Zone / Restricted Vehicular Access**
- **Gate / Pneumatic Bollard / Restricted Access**
- **Electric Vehicle (220V) Charging Stations**
- **Service and Delivery Only**
- **Loading Zone**
- **Truck Route**

INSET 2

- **Vehicle Exclusion Zone**
- **No Parking Zone**

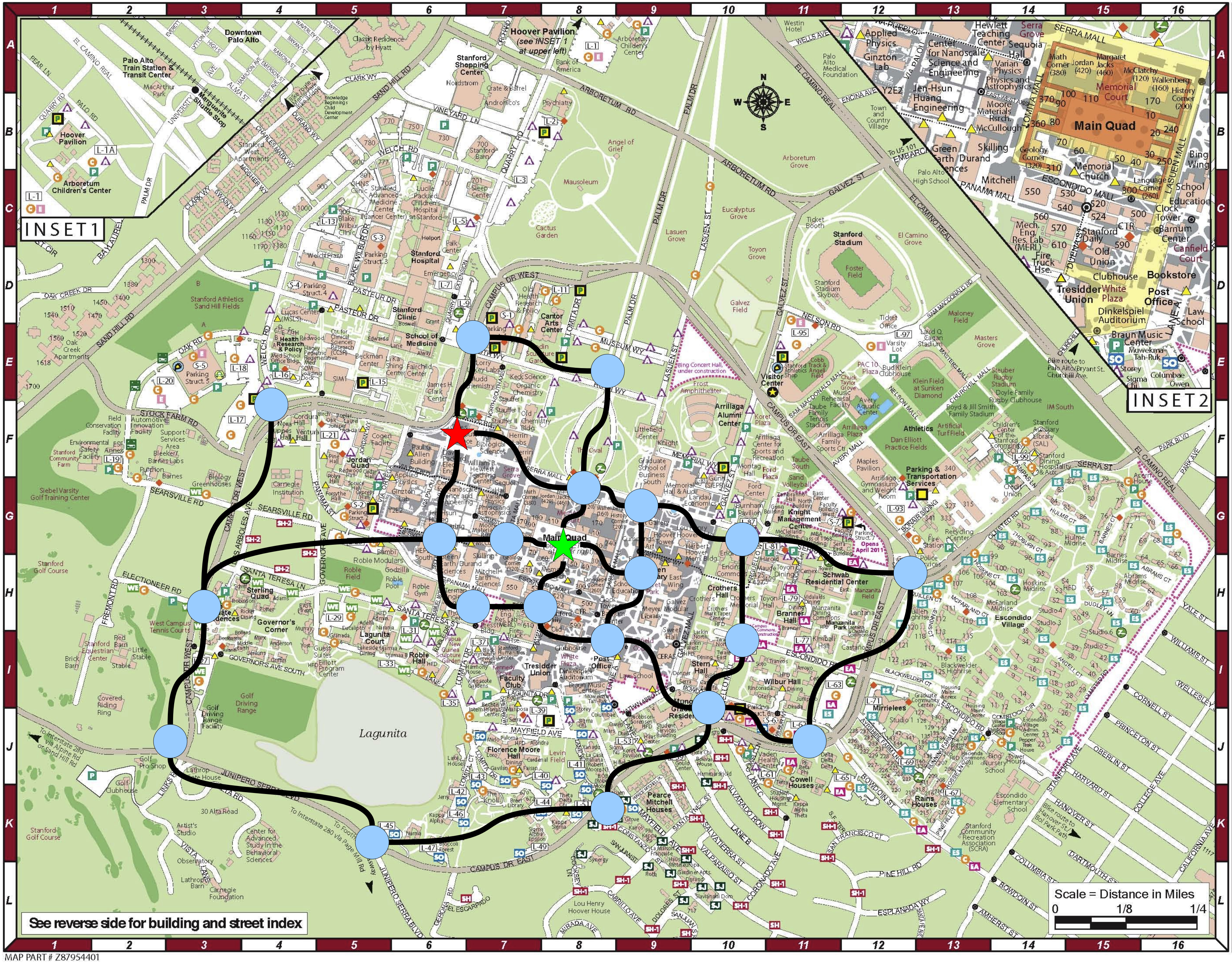
For more information on designated cart routes, cart parking, and other details related to pedestrian zone access, refer to the **Ped Zone Access Info and Map:** *transportation.stanford.edu/parking_info/pedzone.shtml*

Parking and Transportation Info
transportation.stanford.edu
650.723.9362

Truck Route Info and Map
transportation.stanford.edu/images/truck-routes.pdf

University Telephone Operator
650.723.2300

MAP PART # Z87954401



MAP PART # Z87954401

STANFORD UNIVERSITY

PARKING AND CIRCULATION MAP 2010-11

- **Parking & Transportation Services** *Parking permit sales, 7:30am - 5pm (M-F)*
- **Visitor Parking** *Pay meter or time limit, 8am - 4pm (M-F)*
- **Visitor Parking / Pay Station** *Pay at machine, 8am - 4pm (M-F) Pay by credit card (Visa/Mastercard) or cash*
- **Event Parking** *6am - 4pm (M-F)*
- **Bus Parking** *6am - 4pm (M-F)*
- **Motorcycle Parking** *6am - 4pm (M-F)*

Commuter Parking *6am - 4pm (M-F)*

- A Permit
- C Permit

Resident Parking *24 hours, 7 days*

- EA Permit
- SJ Permit
- ES Permit
- SO Permit
- SH Permit
- WE Permit

Many lots are signed only at the entrance. In "shared" lots (e.g., those posted "EA or C") you may park with either permit designated.

Disability Parking

Any State issued disability parking permit is valid in any parking space on campus. There are DP spaces at most buildings. For more info, see maps.stanford.edu/ada

- Parking Lot Number
- Parking Structure Number
- Underground Parking Structure Number
- Visitor Center 650.723.2560
- Car Rental
- Car-Sharing Vehicle
- Athletic Fields
- Construction
- Pedestrian Zone / Restricted Vehicular Access
- Gate / Pneumatic Bollard / Restricted Access
- Electric Vehicle (220V) Charging Stations
- Service and Delivery Only
- Loading Zone
- Truck Route

INSET 2

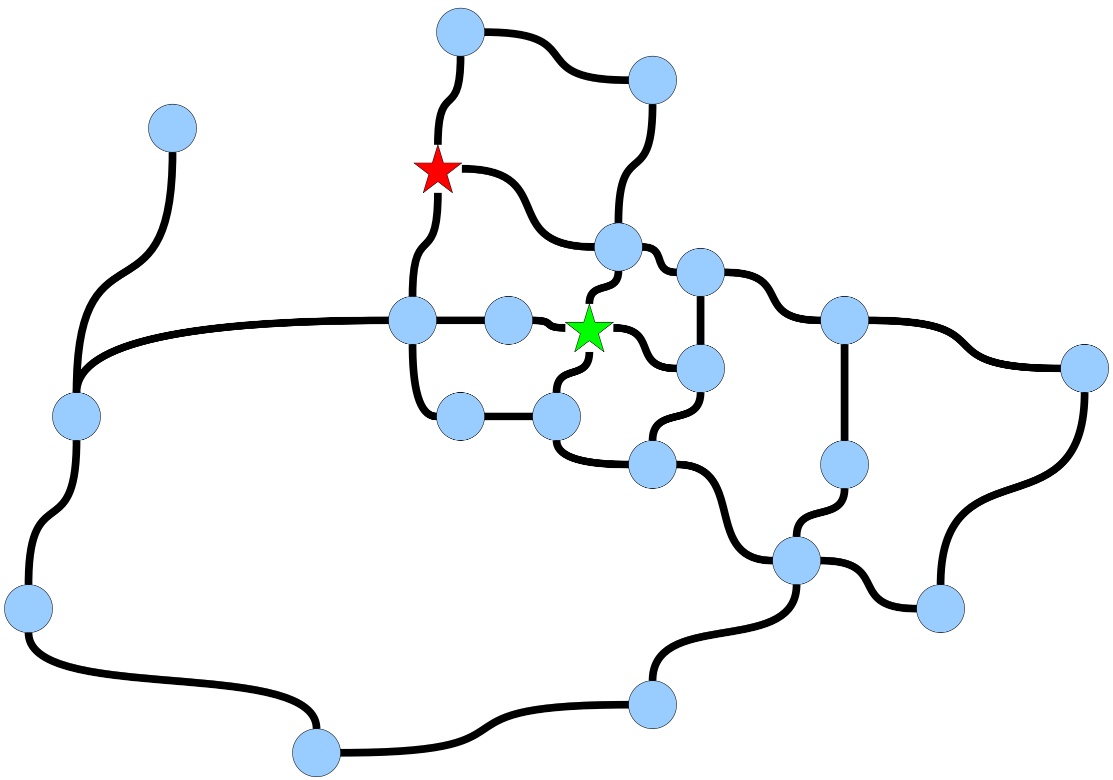
- Vehicle Exclusion Zone
- No Parking Zone

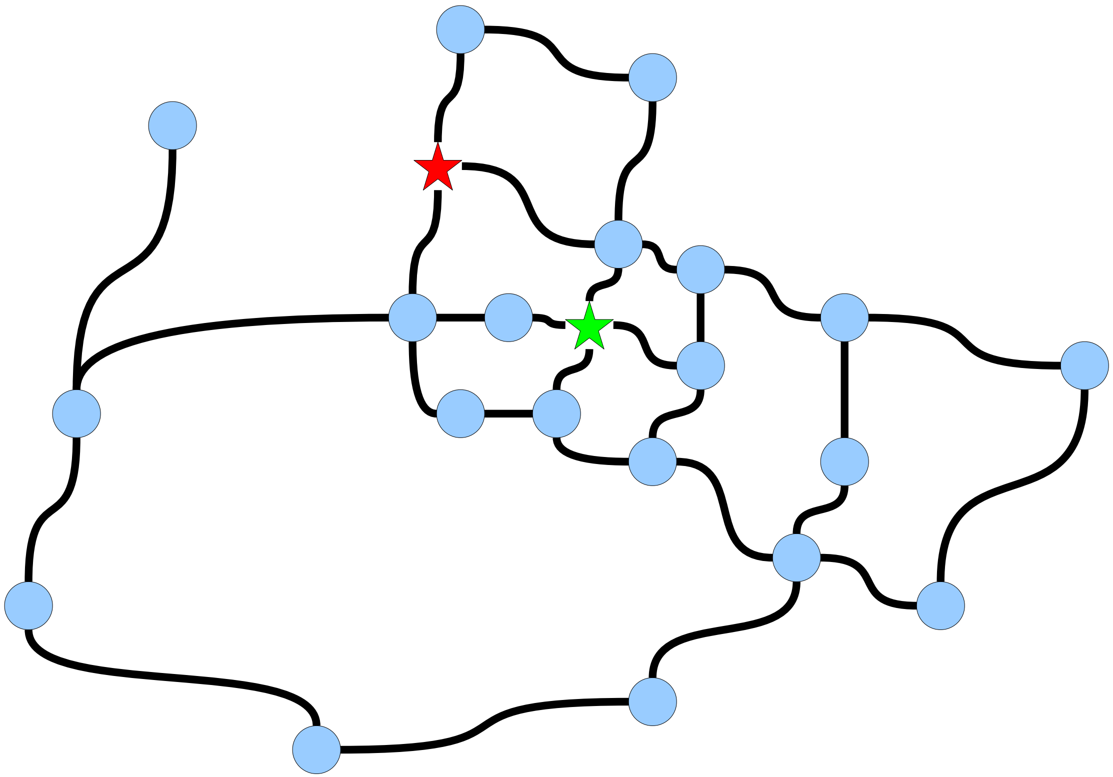
For more information on designated cart routes, cart parking, and other details related to pedestrian zone access, refer to the **Ped Zone Access Info and Map:** *transportation.stanford.edu/parking_info/pedzone.shtml*

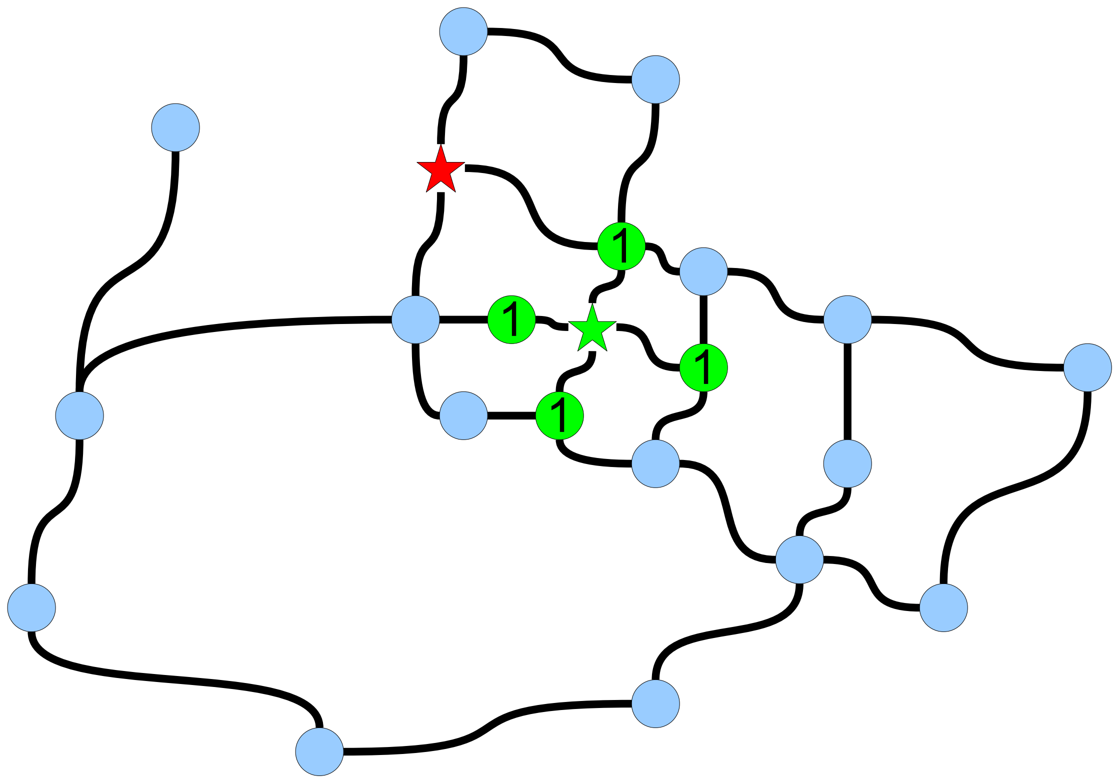
Parking and Transportation Info
transportation.stanford.edu
650.723.9362

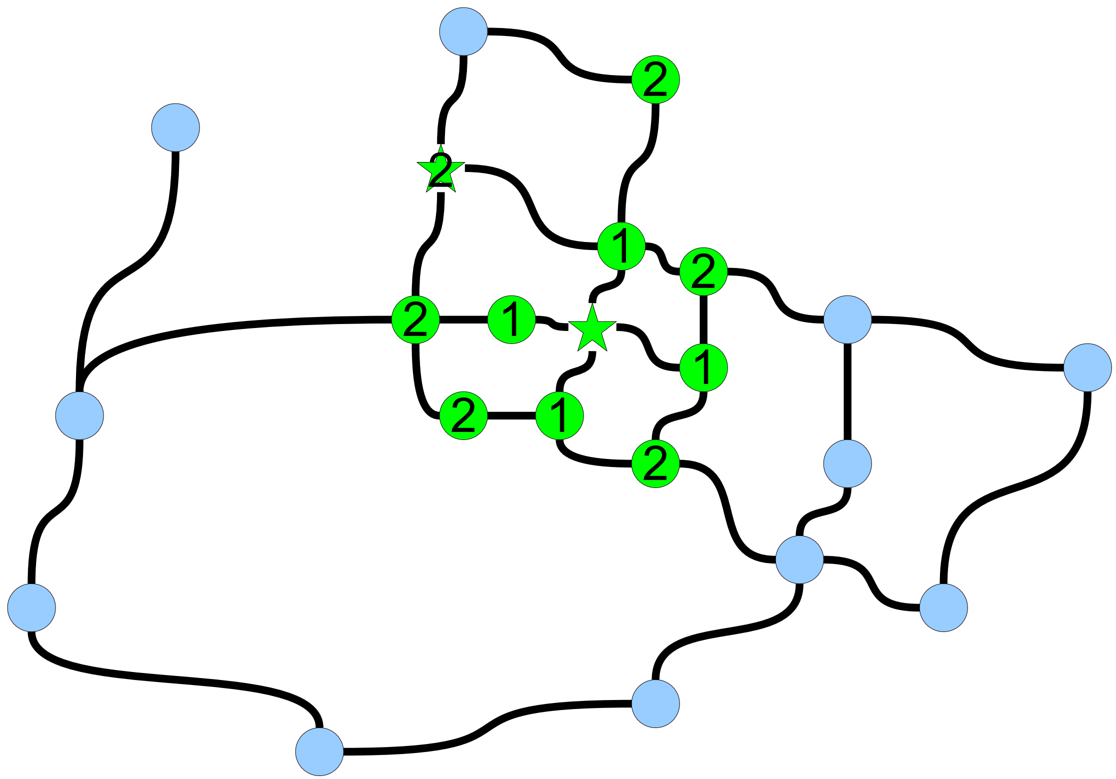
Truck Route Info and Map
transportation.stanford.edu/images/truck-routes.pdf

University Telephone Operator
650.723.2300

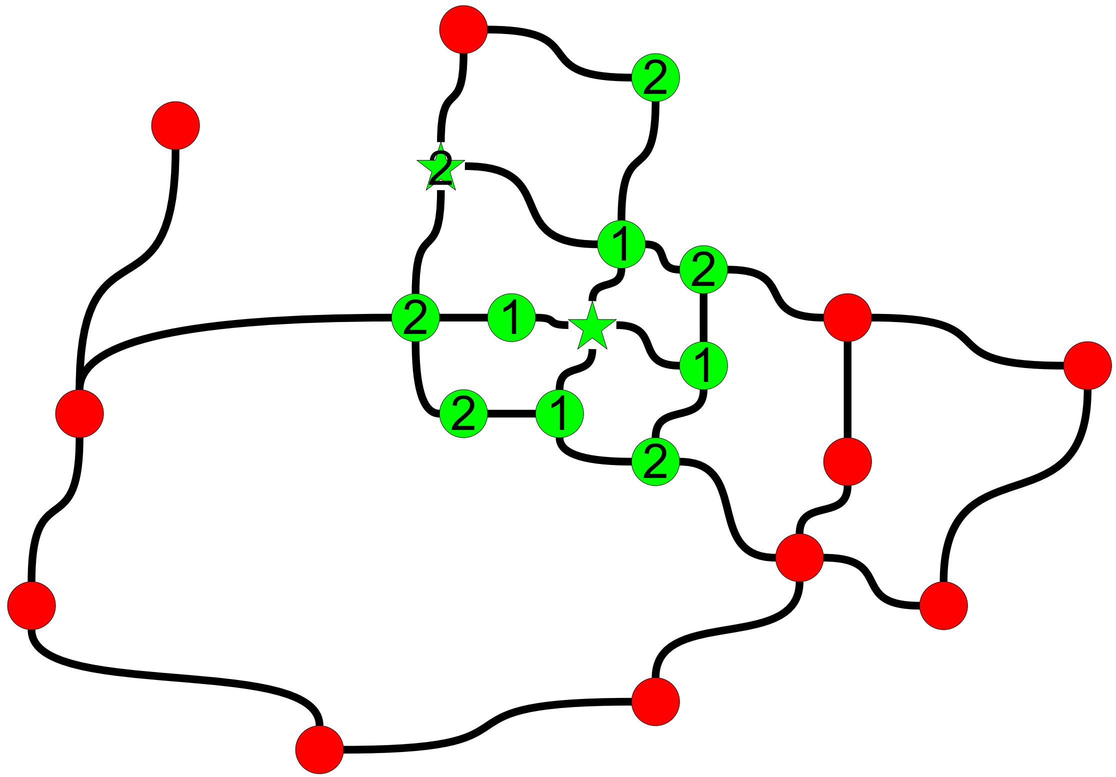




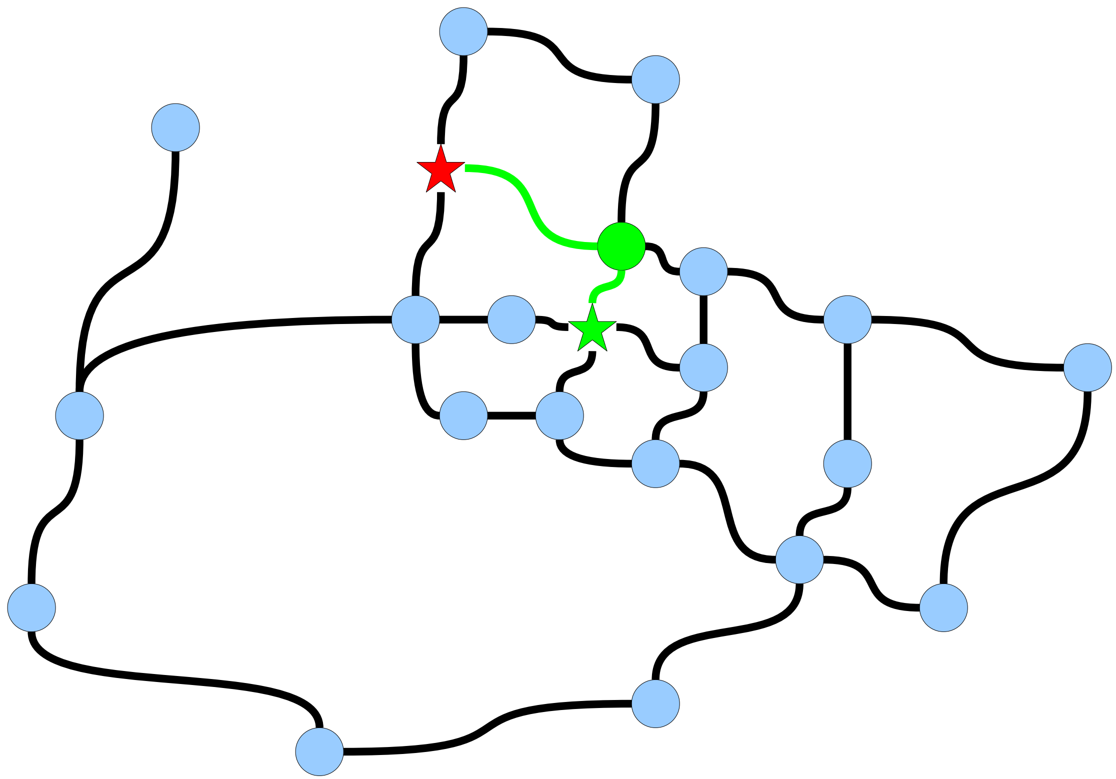

1
1
1
1



2
2
1
2
2
1
1
1
2
2



2
2
1
2
2
1
1
2
1
2

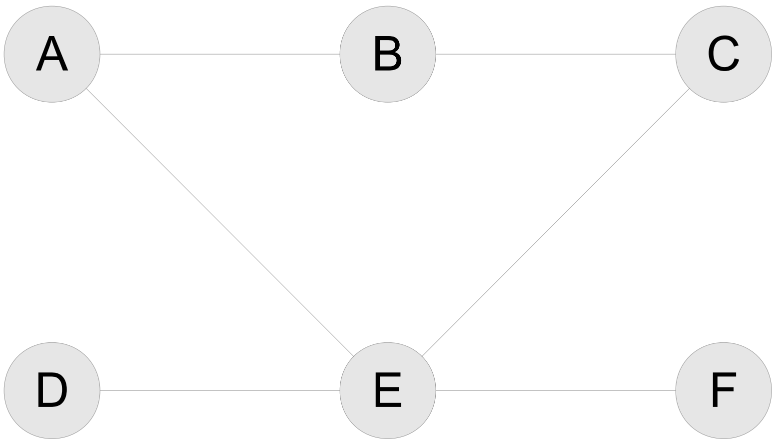


Breadth-First Search

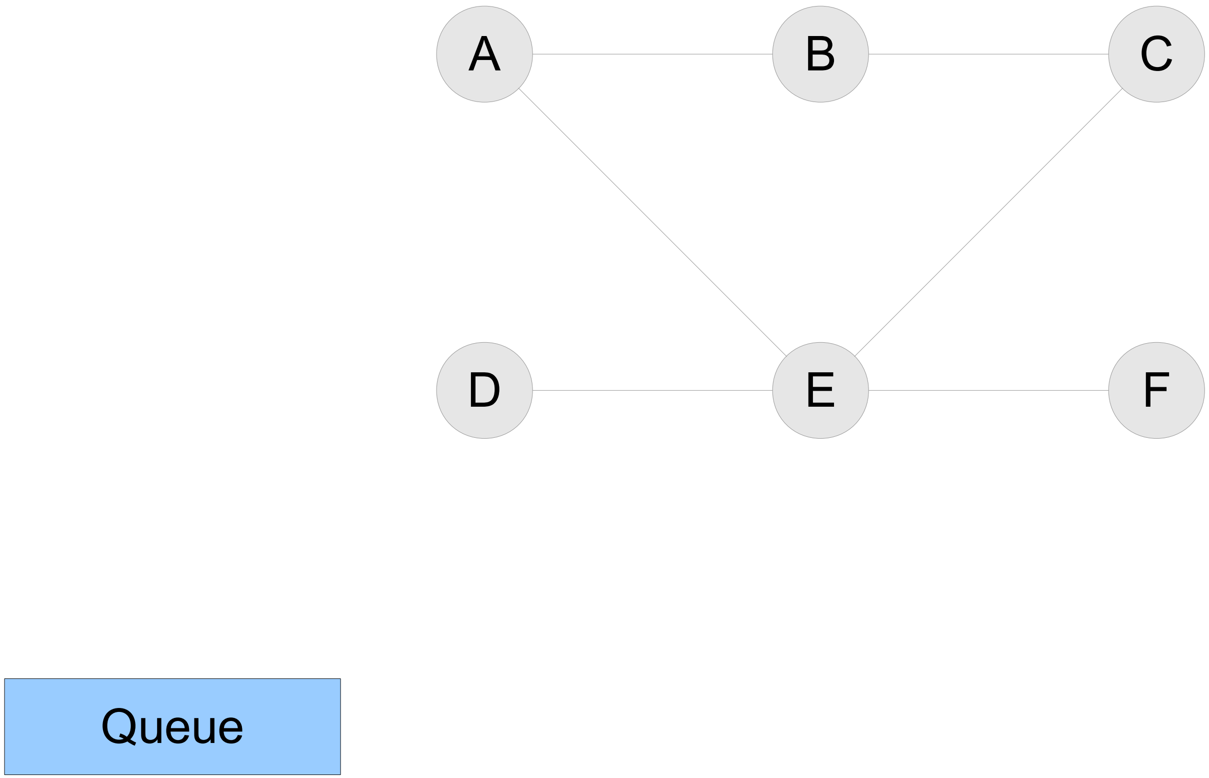
- Specialization of the general search algorithm where nodes to visit are put into a *queue*.
- Explores nodes one hop away, then two hops away, etc.
- Finds path with fewest edges from start node to all other nodes.

Note: The following animation has been simplified for pedagogic purposes. In reality there would be a **`Set`** keeping track of visited nodes and redundant adds to the **`Queue`**

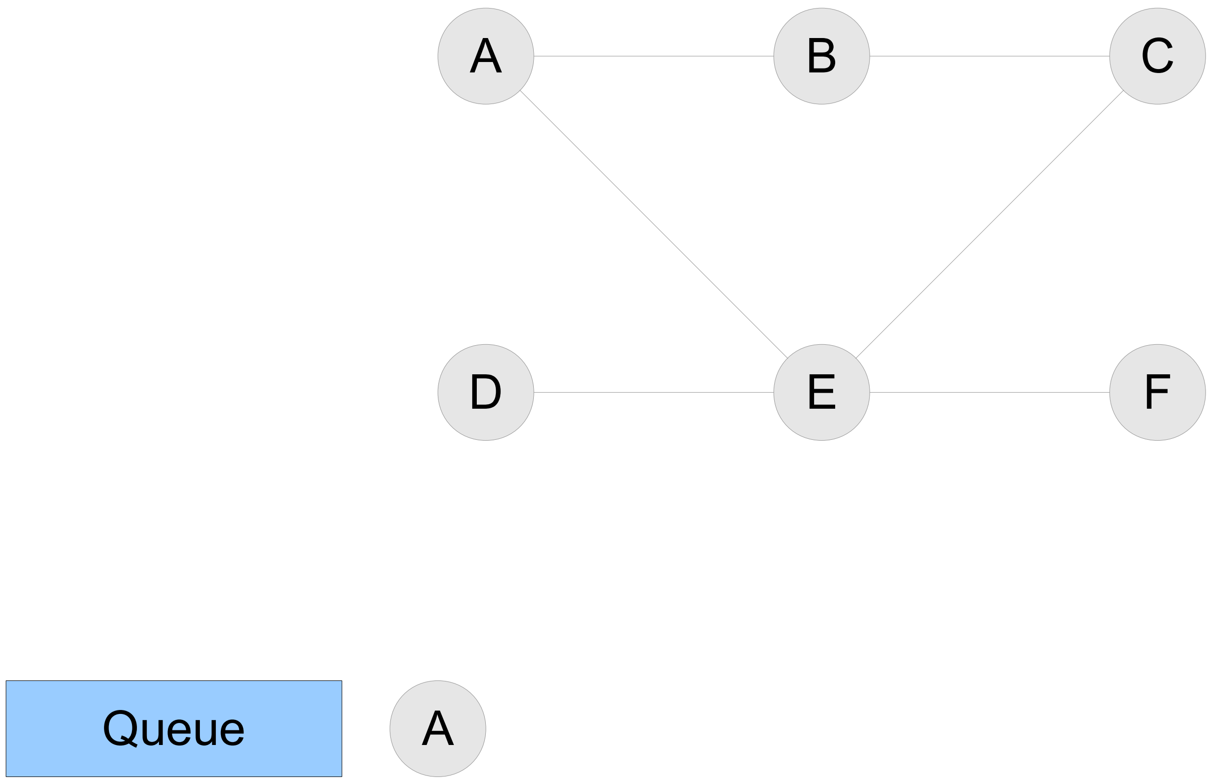
Breadth-first search



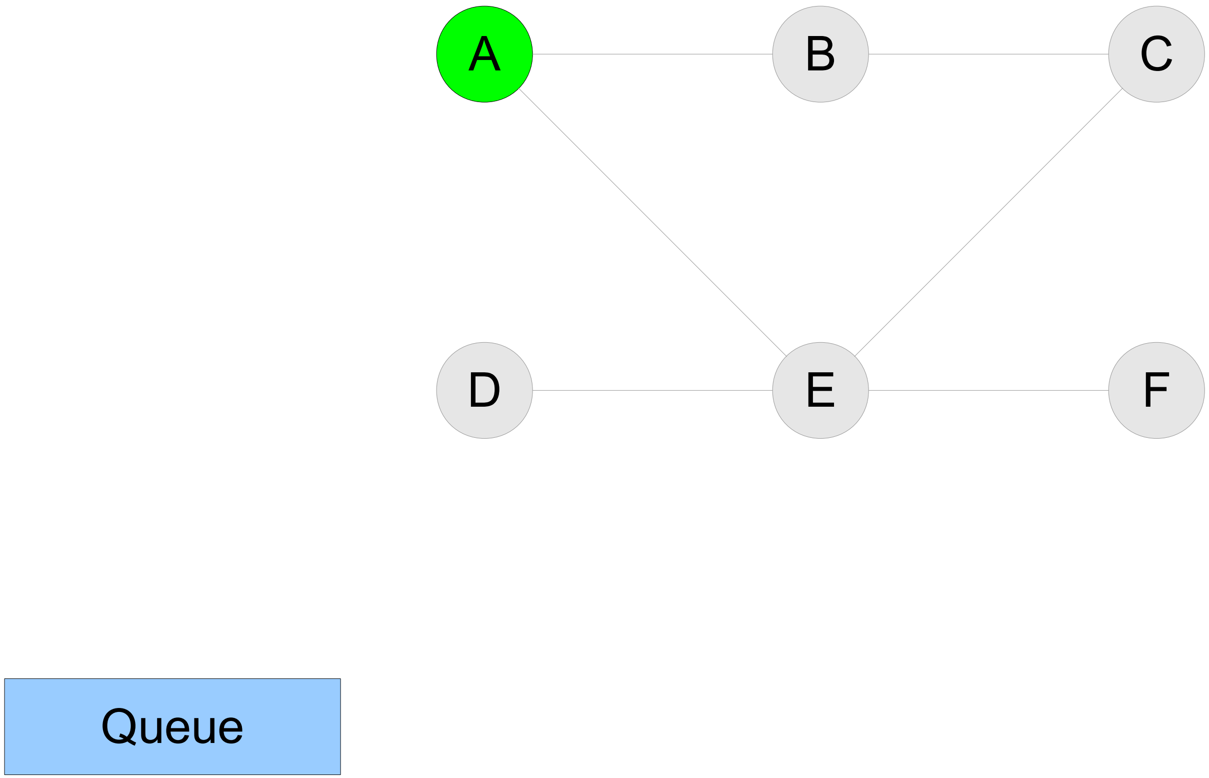
Breadth-first search



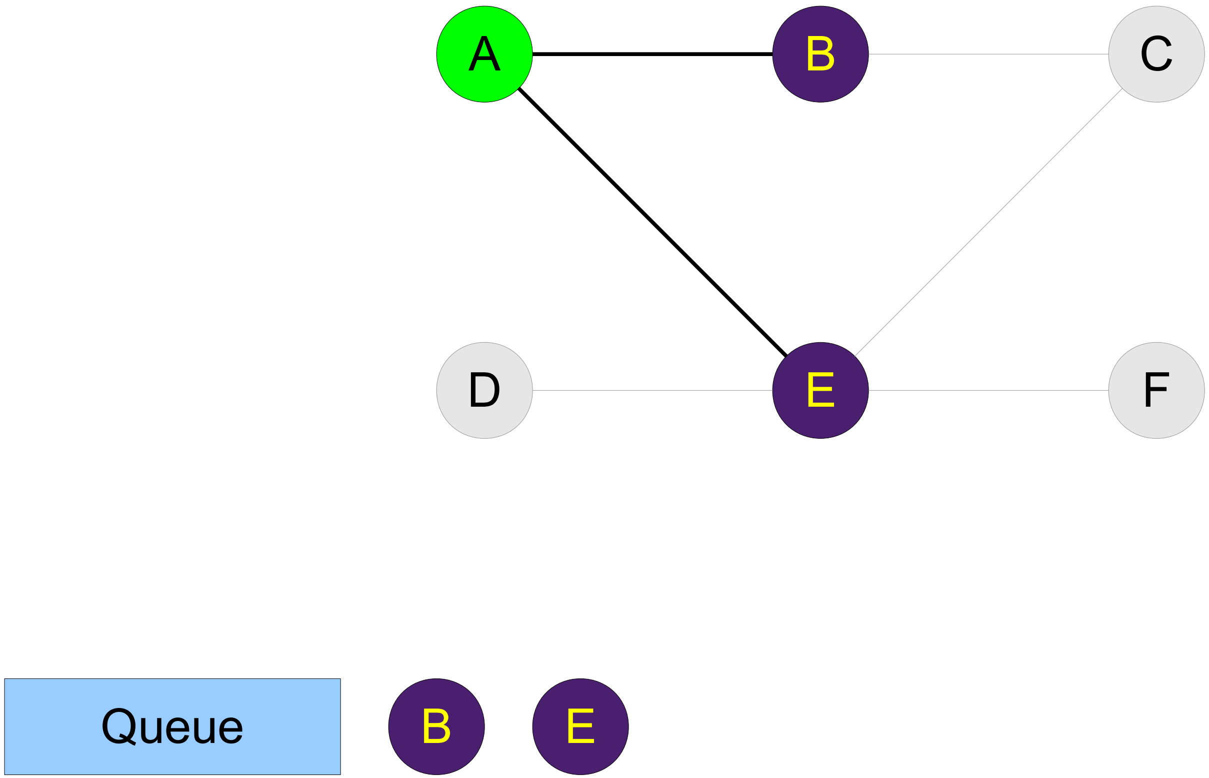
Breadth-first search



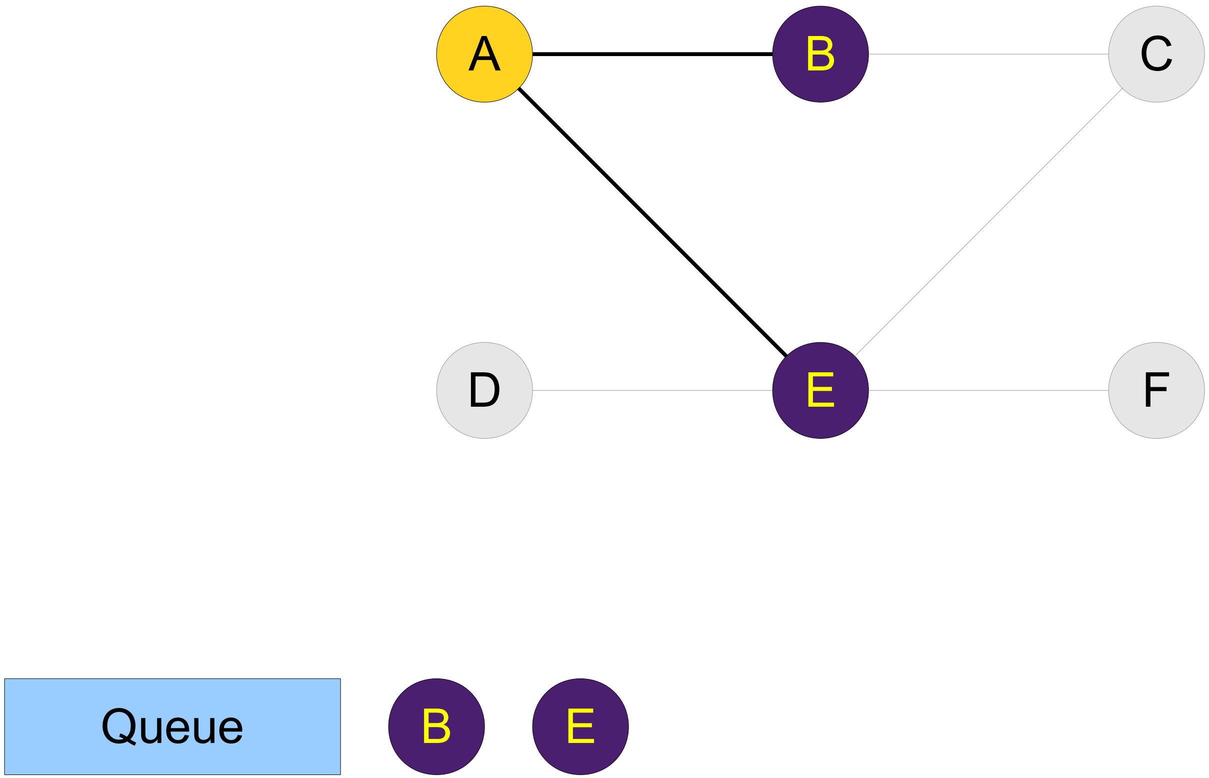
Breadth-first search



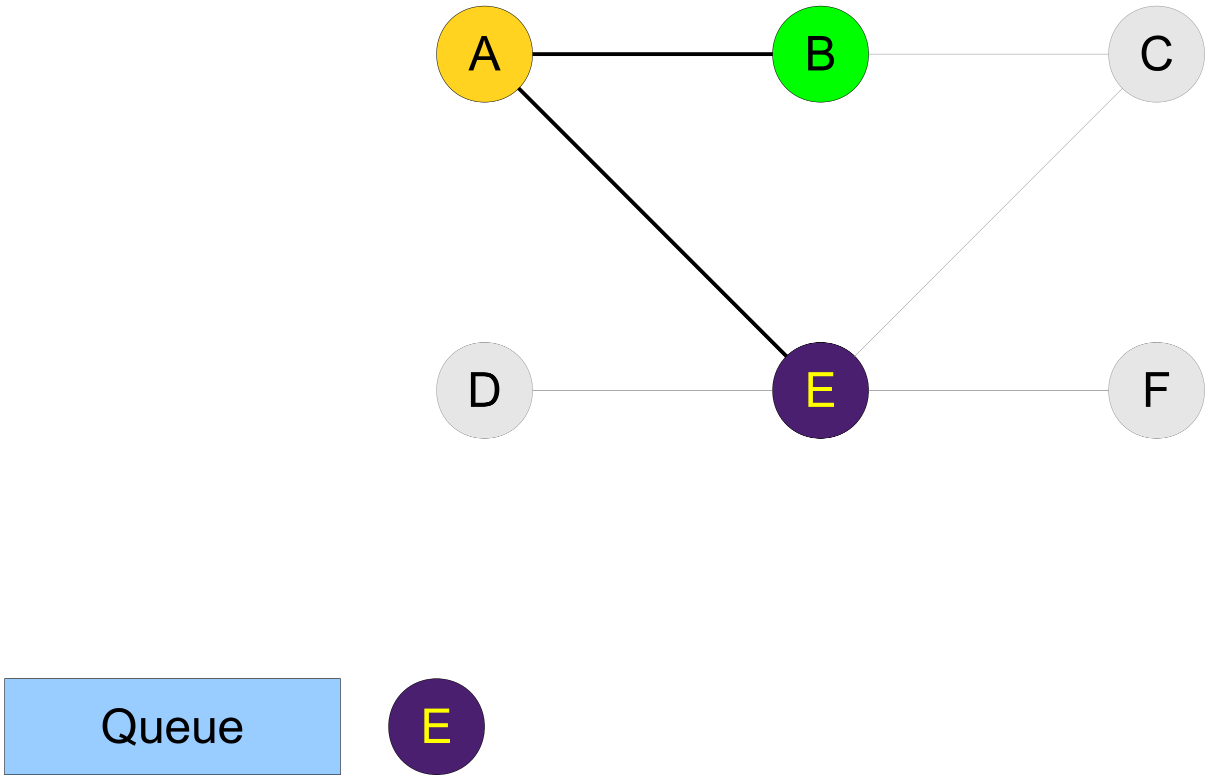
Breadth-first search



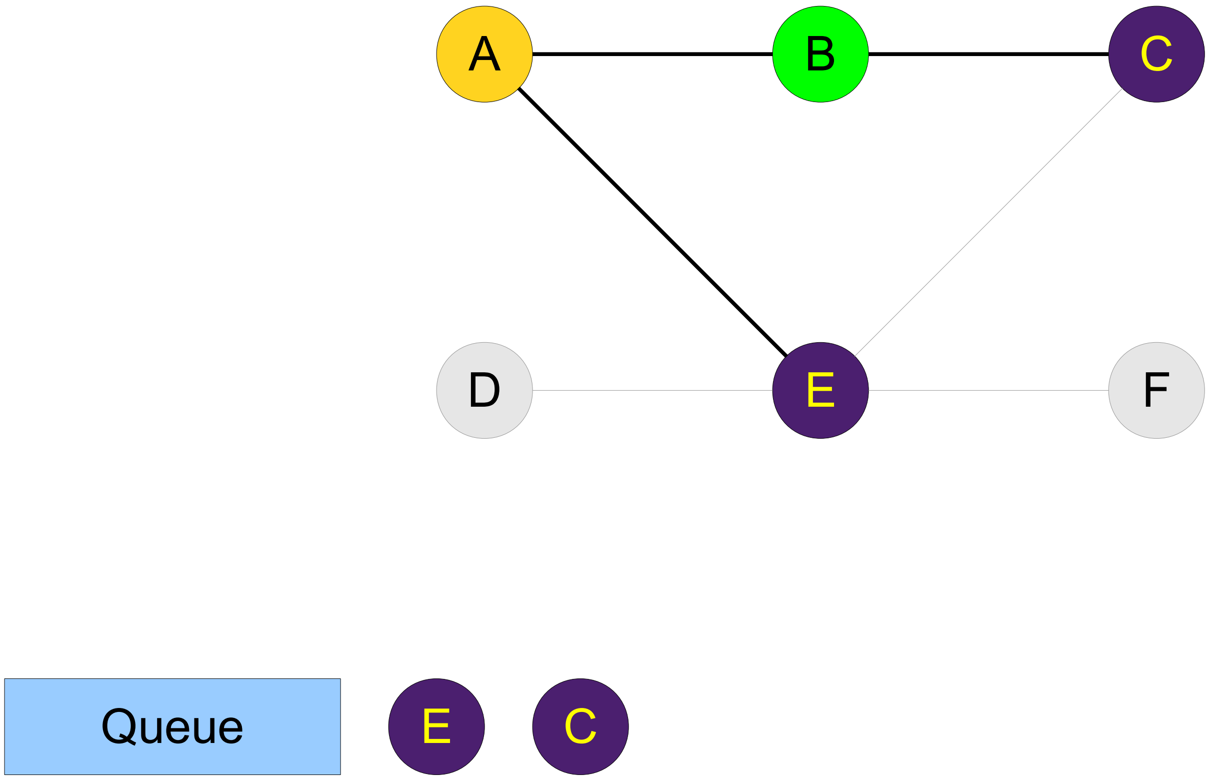
Breadth-first search



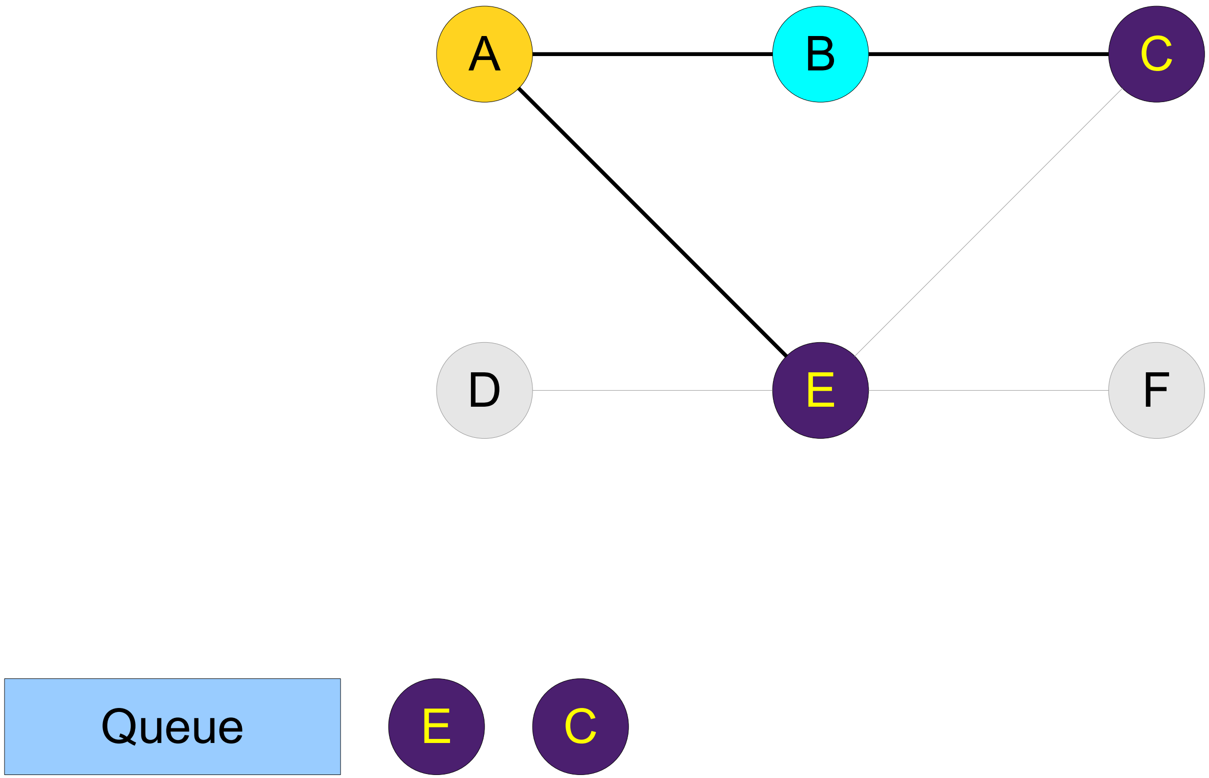
Breadth-first search



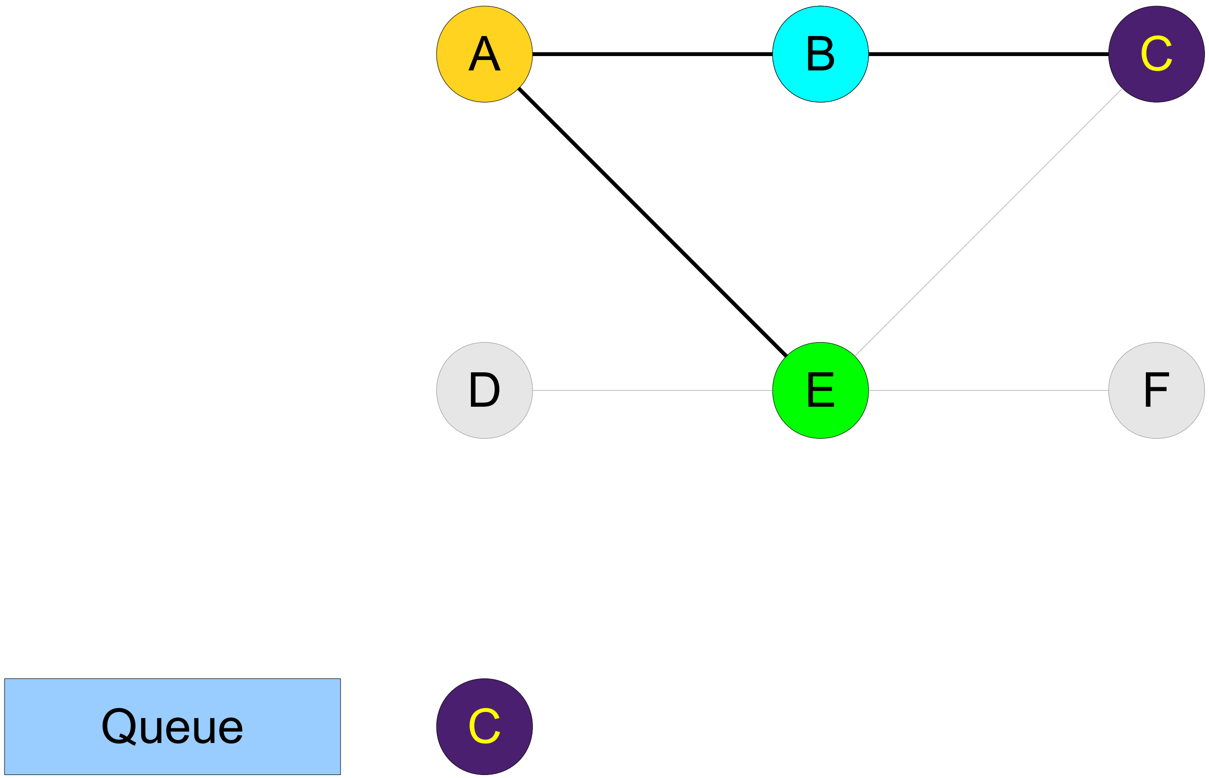
Breadth-first search



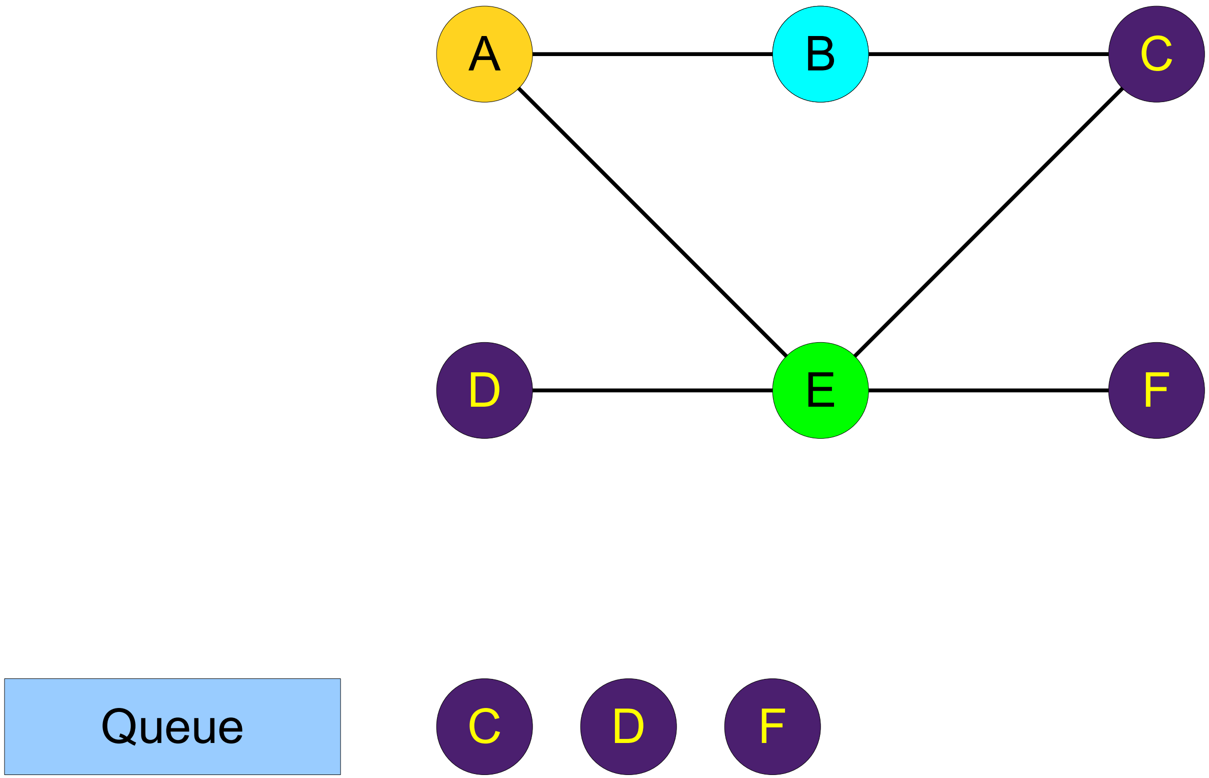
Breadth-first search



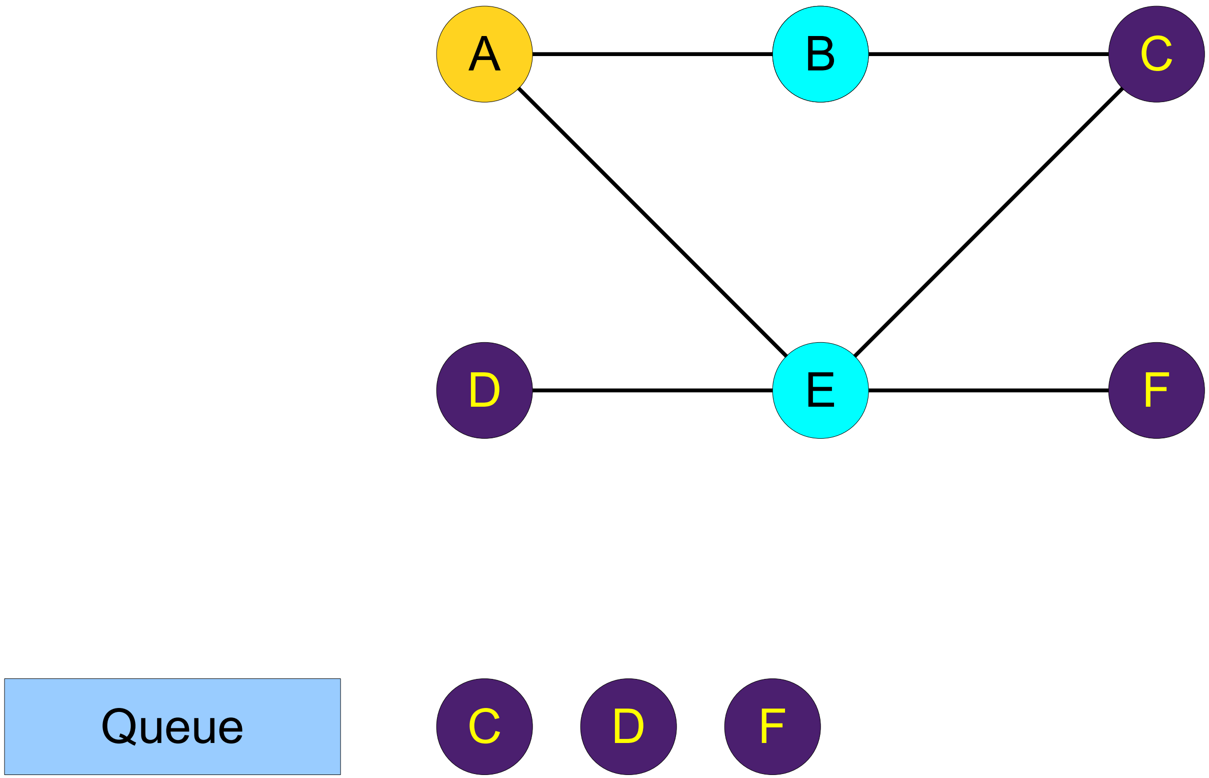
Breadth-first search



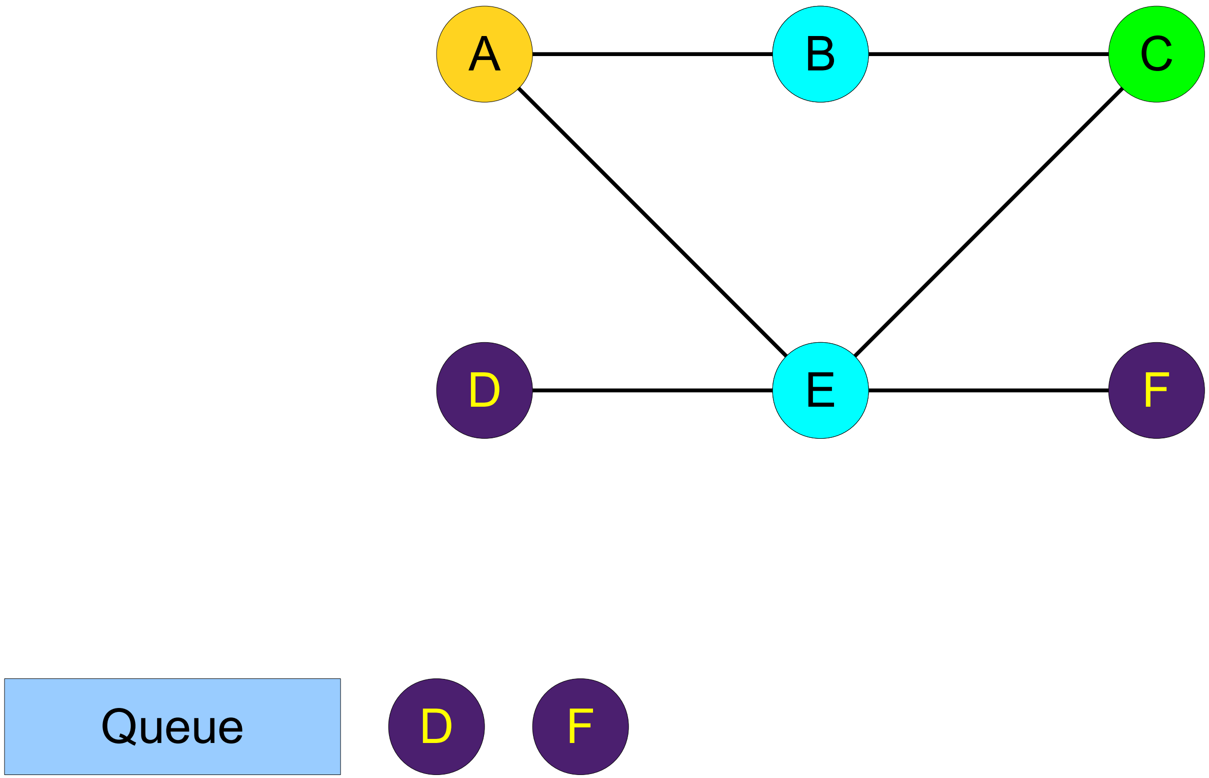
Breadth-first search



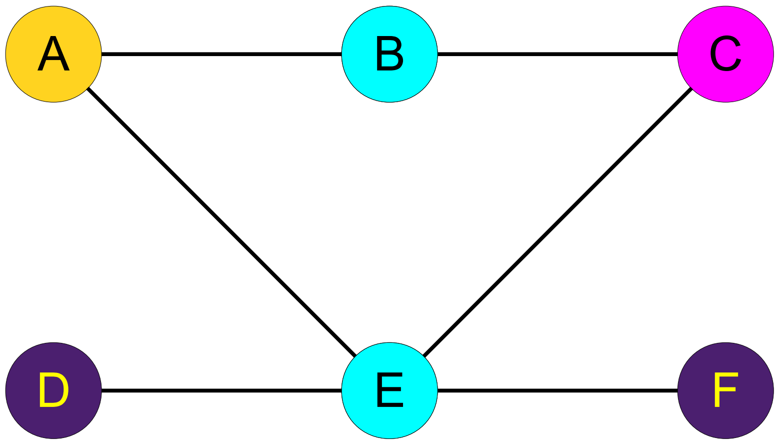
Breadth-first search

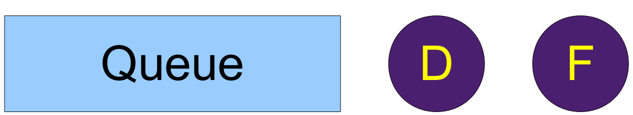


Breadth-first search

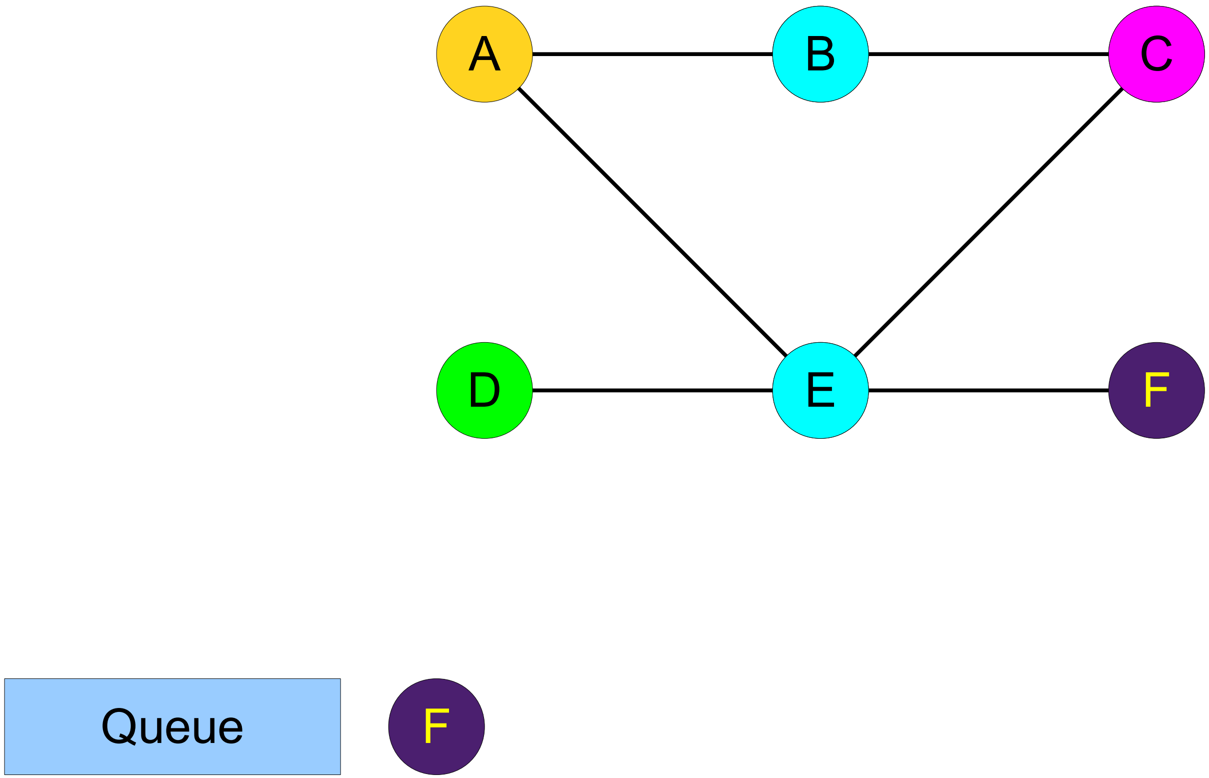


Breadth-first search

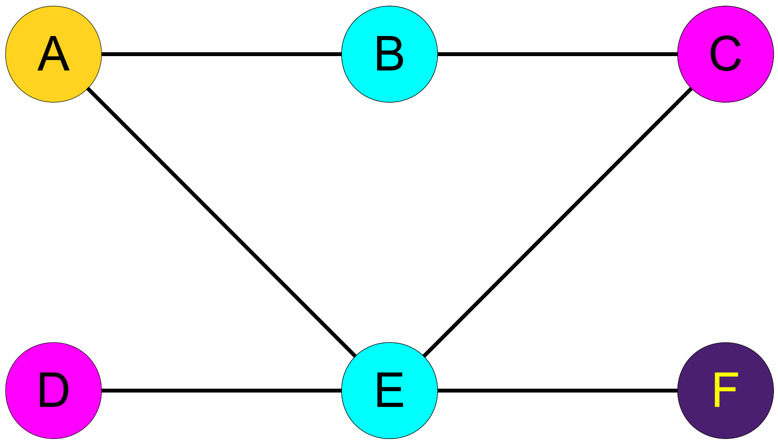


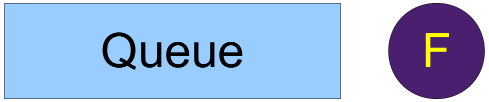


Breadth-first search

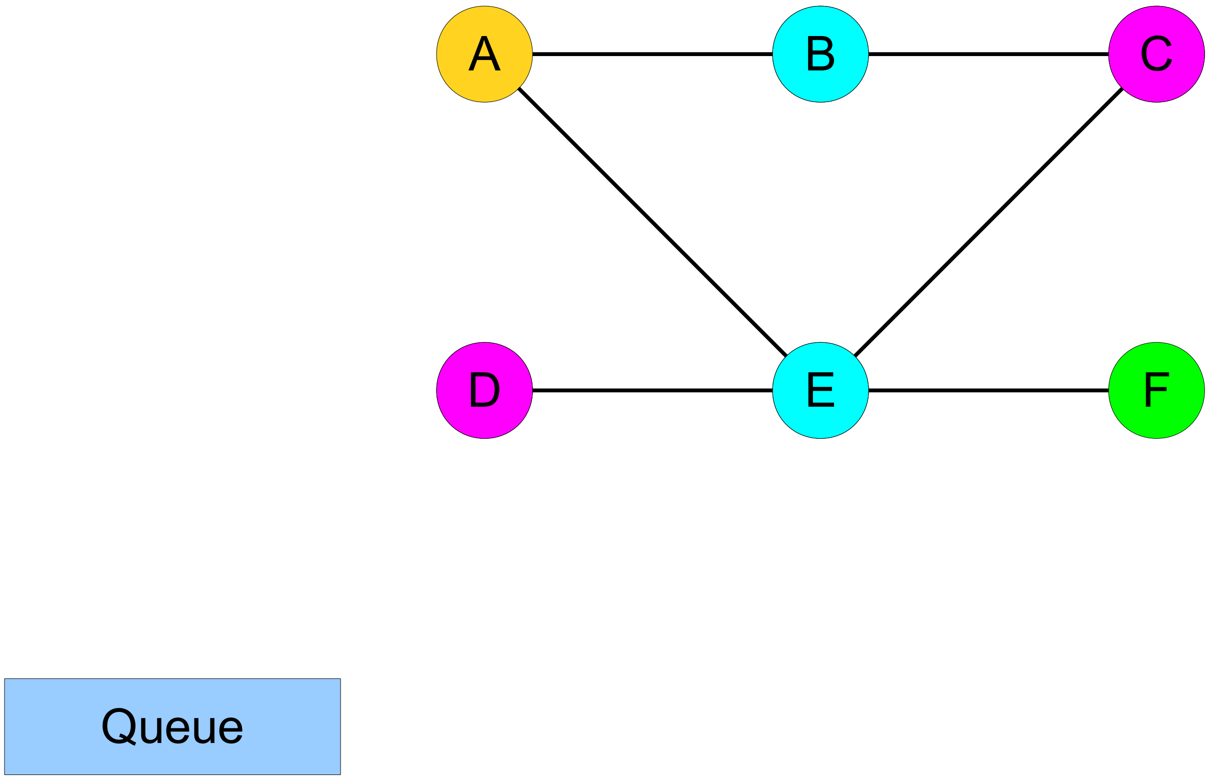


Breadth-first search

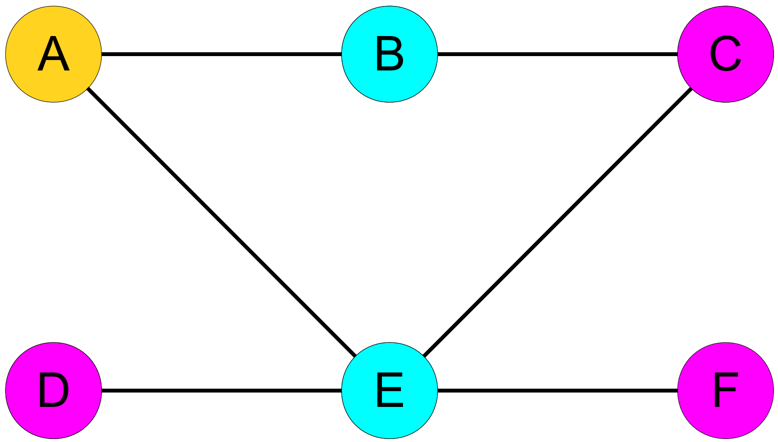




Breadth-first search

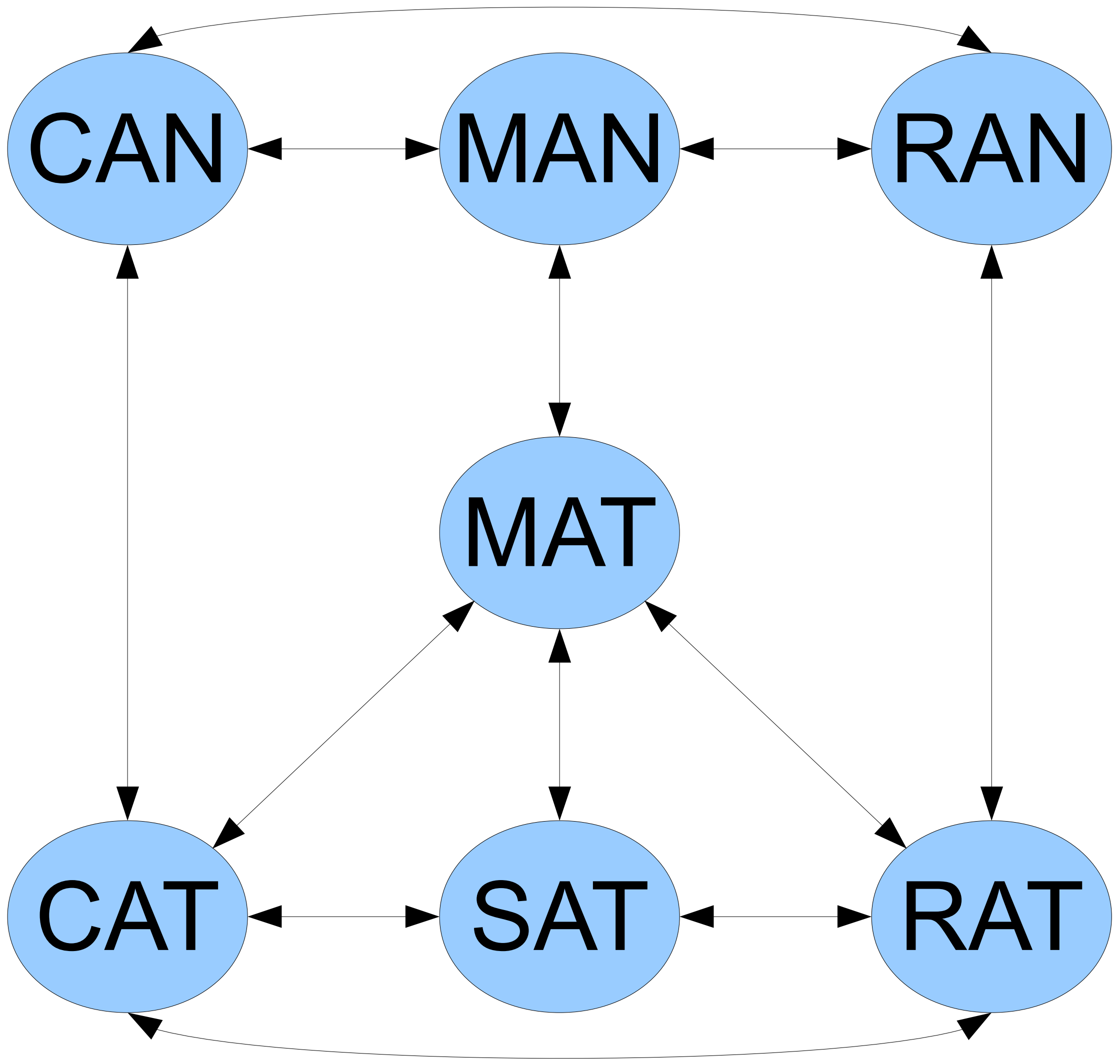


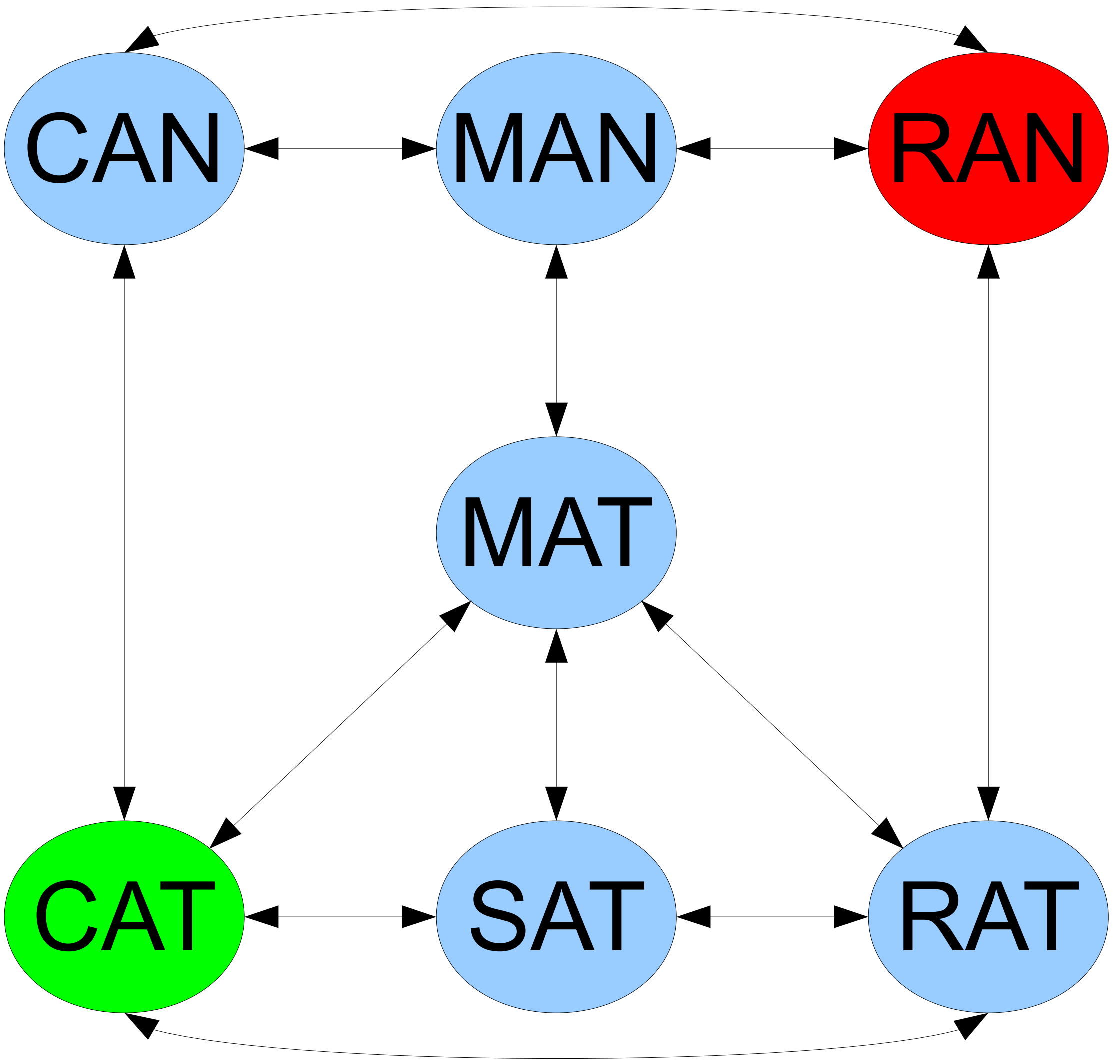
Breadth-first search



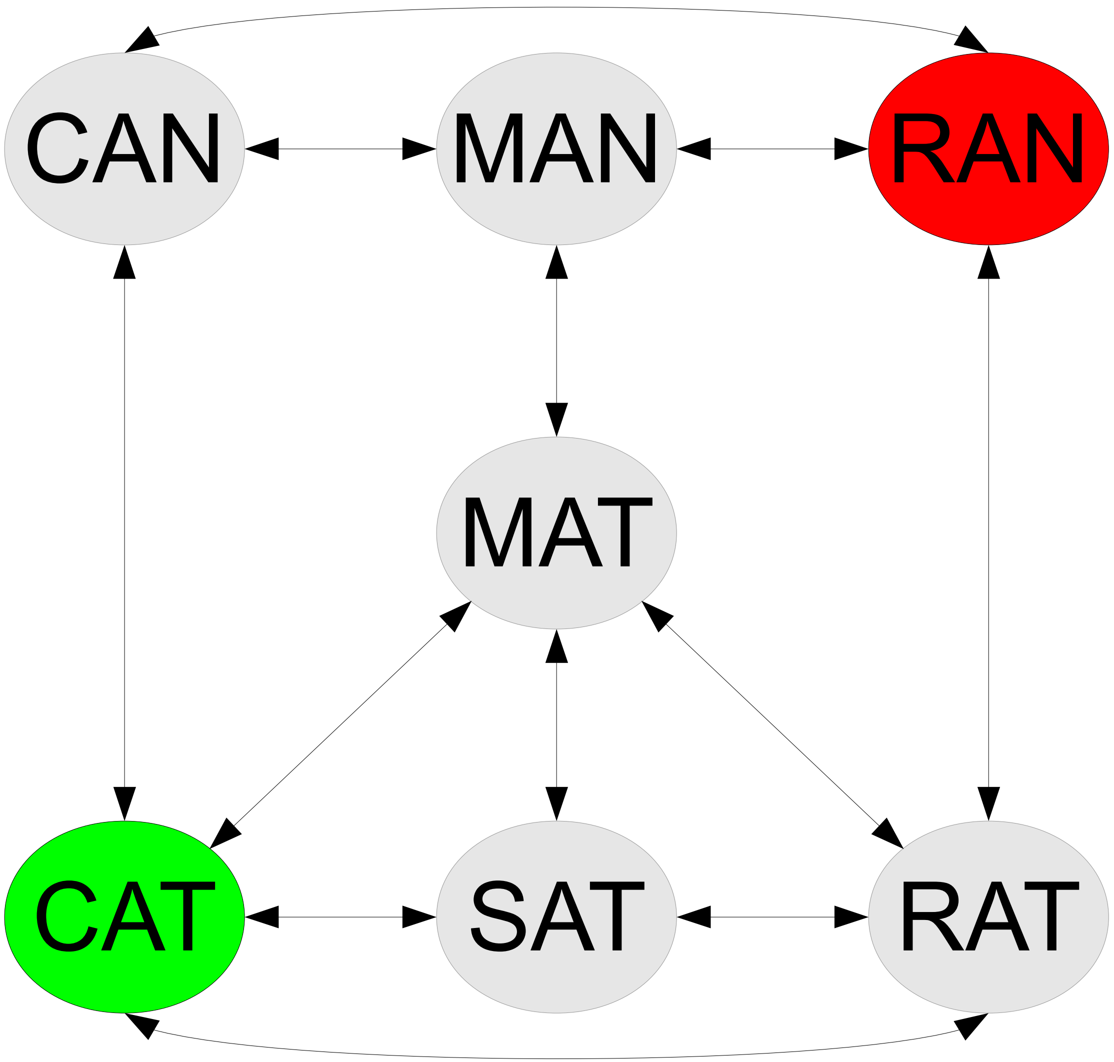
Queue

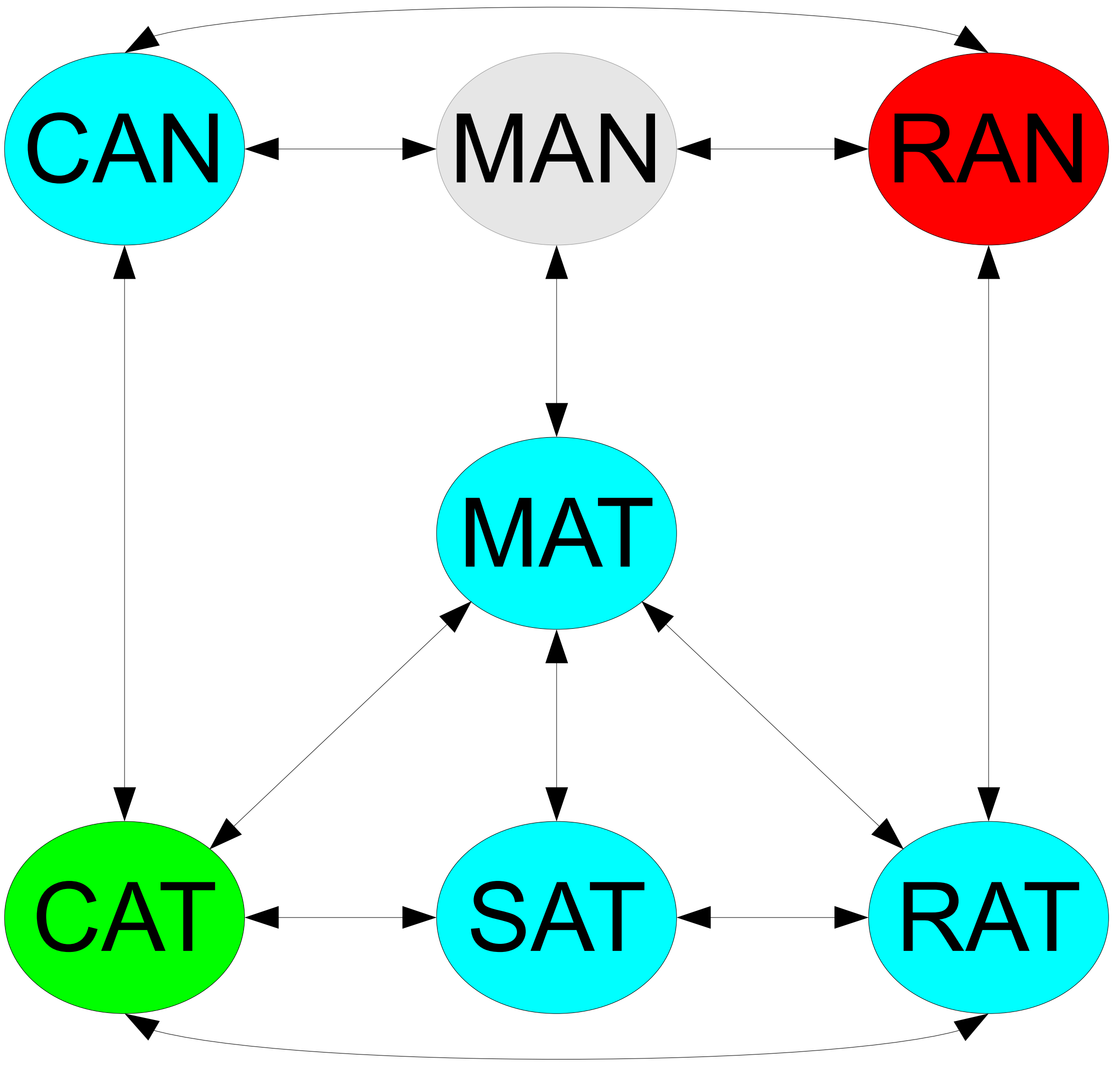
Coding Breadth-First Search

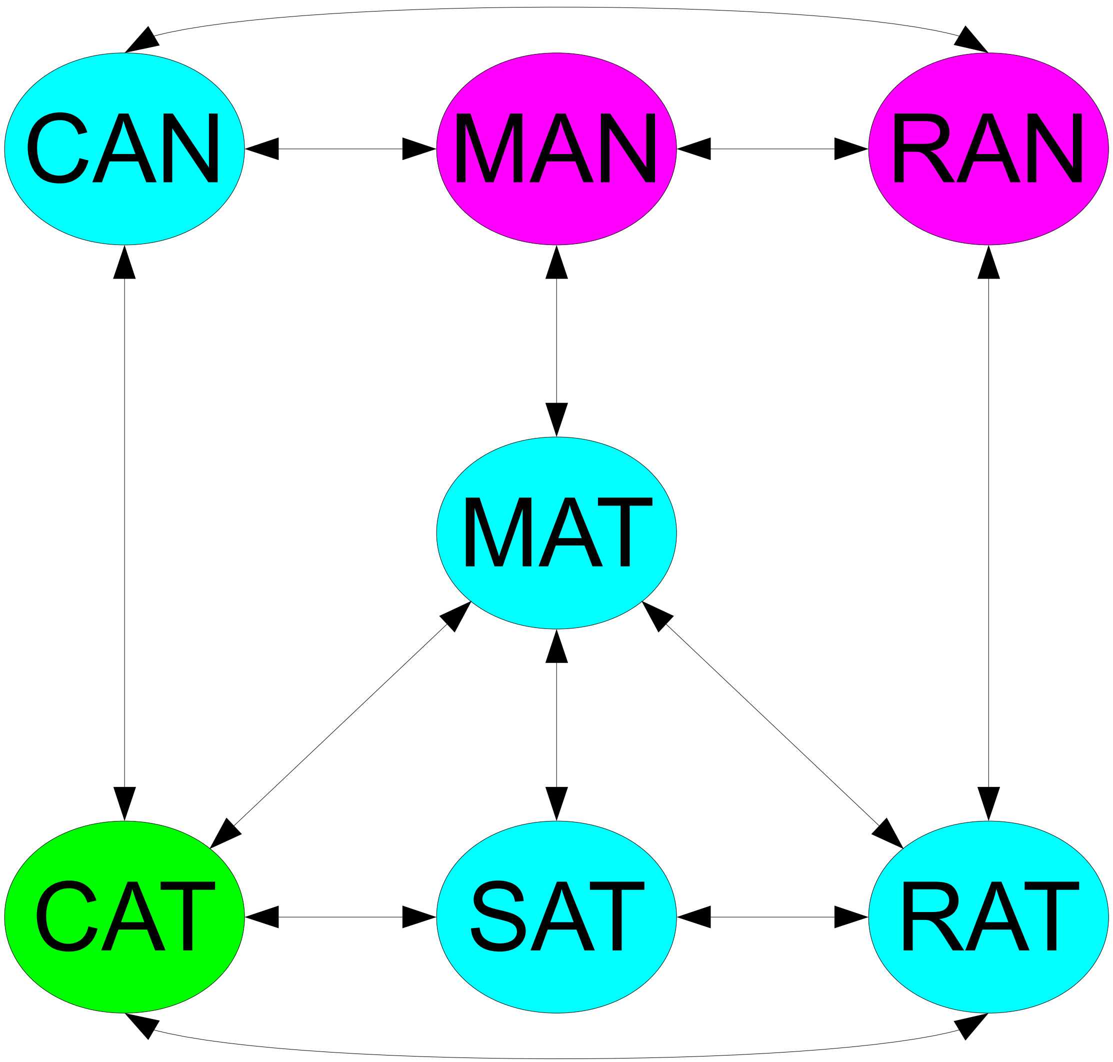

CAN
MAN
RAN
MAT
CAT
SAT
RAT

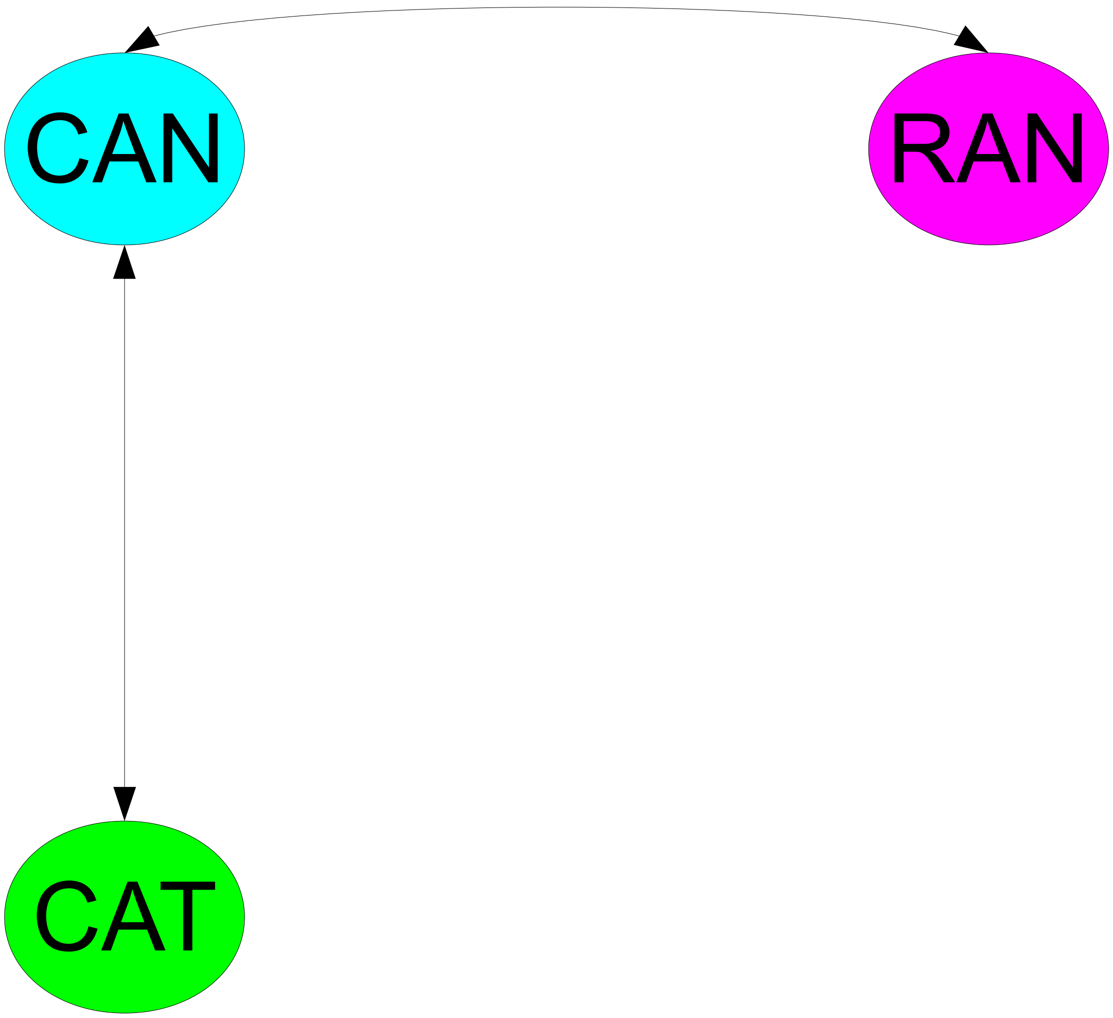


CAN
MAN
RAN
MAT
CAT
SAT
RAT


CAN
MAN
RAN
MAT
CAT
SAT
RAT


CAN
MAN
RAN
MAT
CAT
SAT
RAT


CAN
MAN
RAN
MAT
CAT
SAT
RAT


CAN
RAN
CAT

Next Time

- **Shortest Paths**
  - Dijkstra's Algorithm.
  - A* Search.