

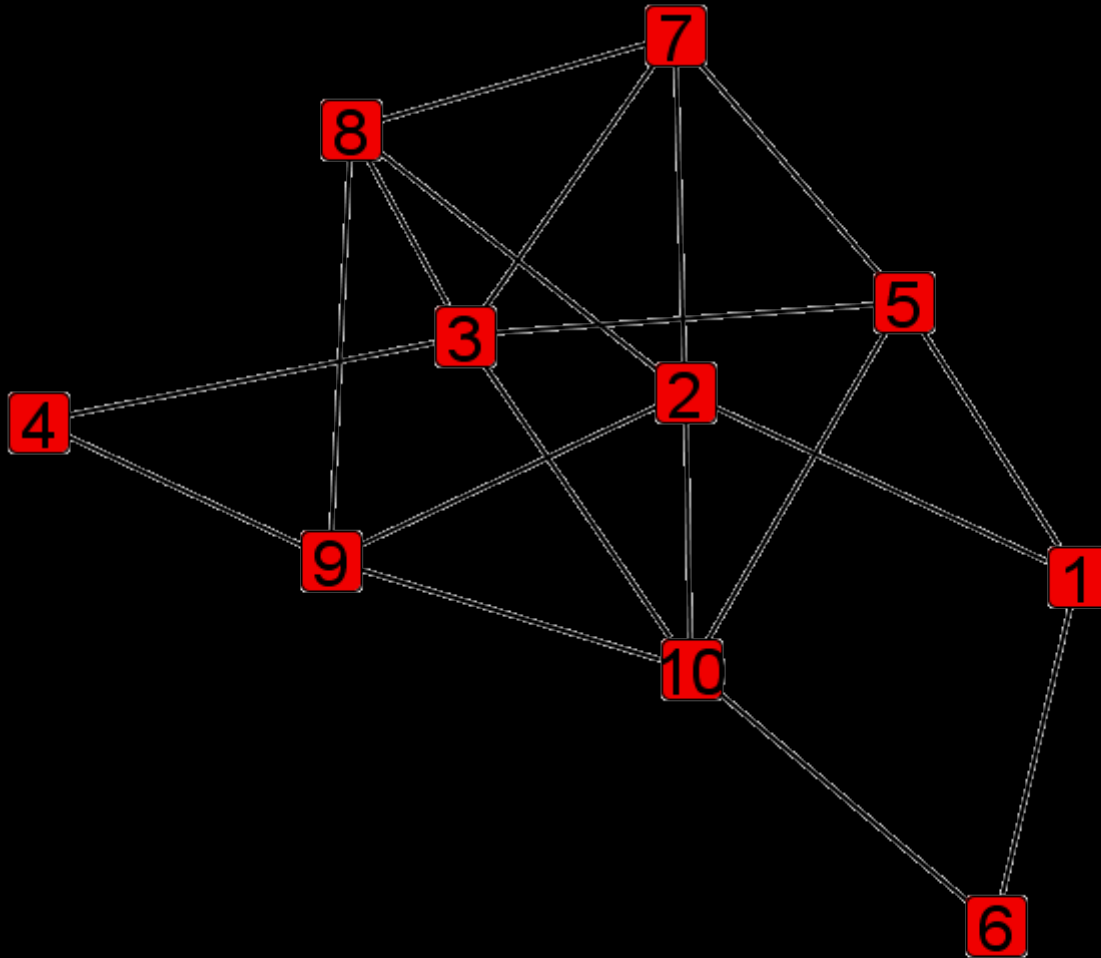
# CS-103 Announcements

- **Midterm in class Fri Oct 27**
  - ❑ 15% of total class grade
  - ❑ Based on lectures up to and including Mon Oct 23—focusing on Oct more than Sept
  - ❑ No HTML
- **New class notes posted to CS103.pbwiki**
- **Reading**
  - **Six Degrees Ch 5 (Searching networks)**
  - **Web 101 Ch 5.3 (Javascript intro)**

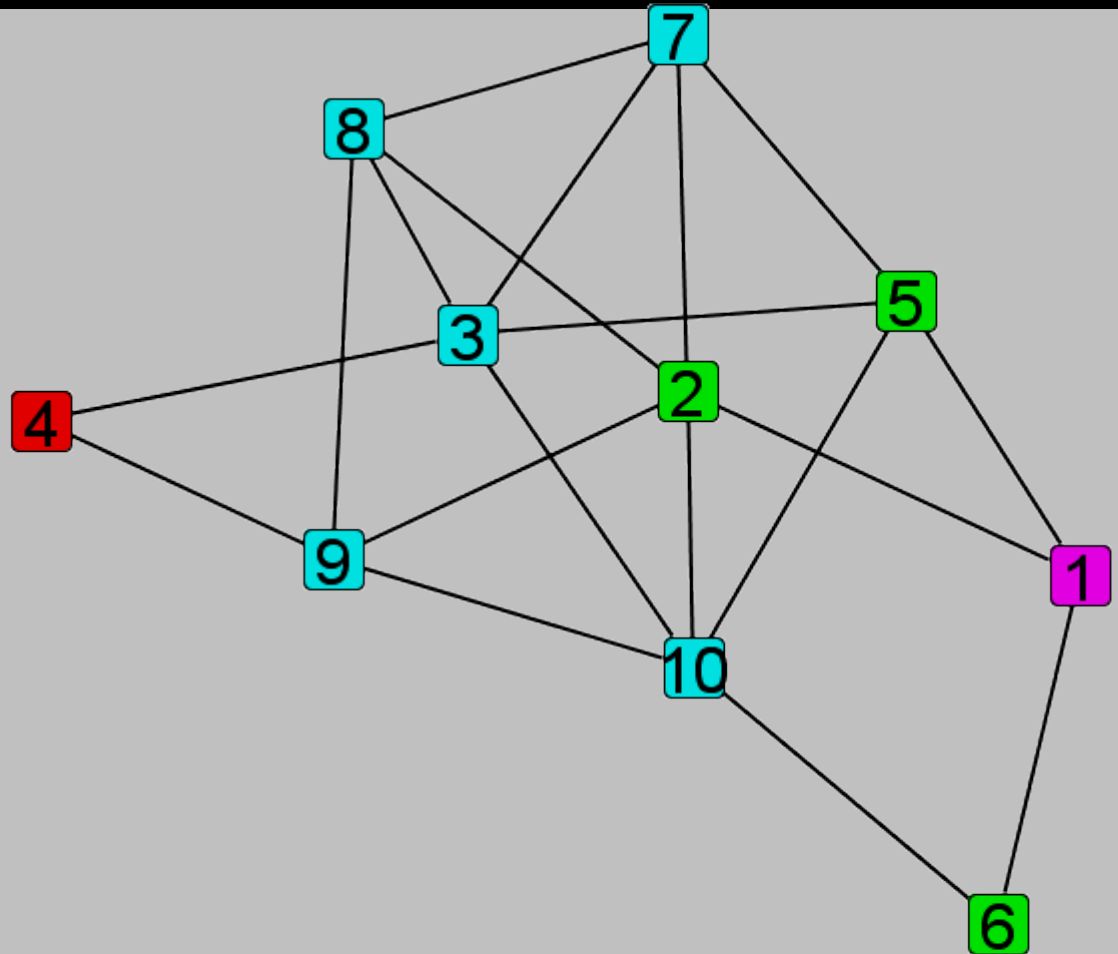
# Midterm Practice Problems

- In lieu of HW, this week we have practice problems for the midterm.
- We will review for midterm and discuss problems in lecture Wed Oct 25
- You may submit solutions before lecture for extra credit. No extra credit after lecture starts.
- No penalty for not submitting solutions

# How can we find shortest path?



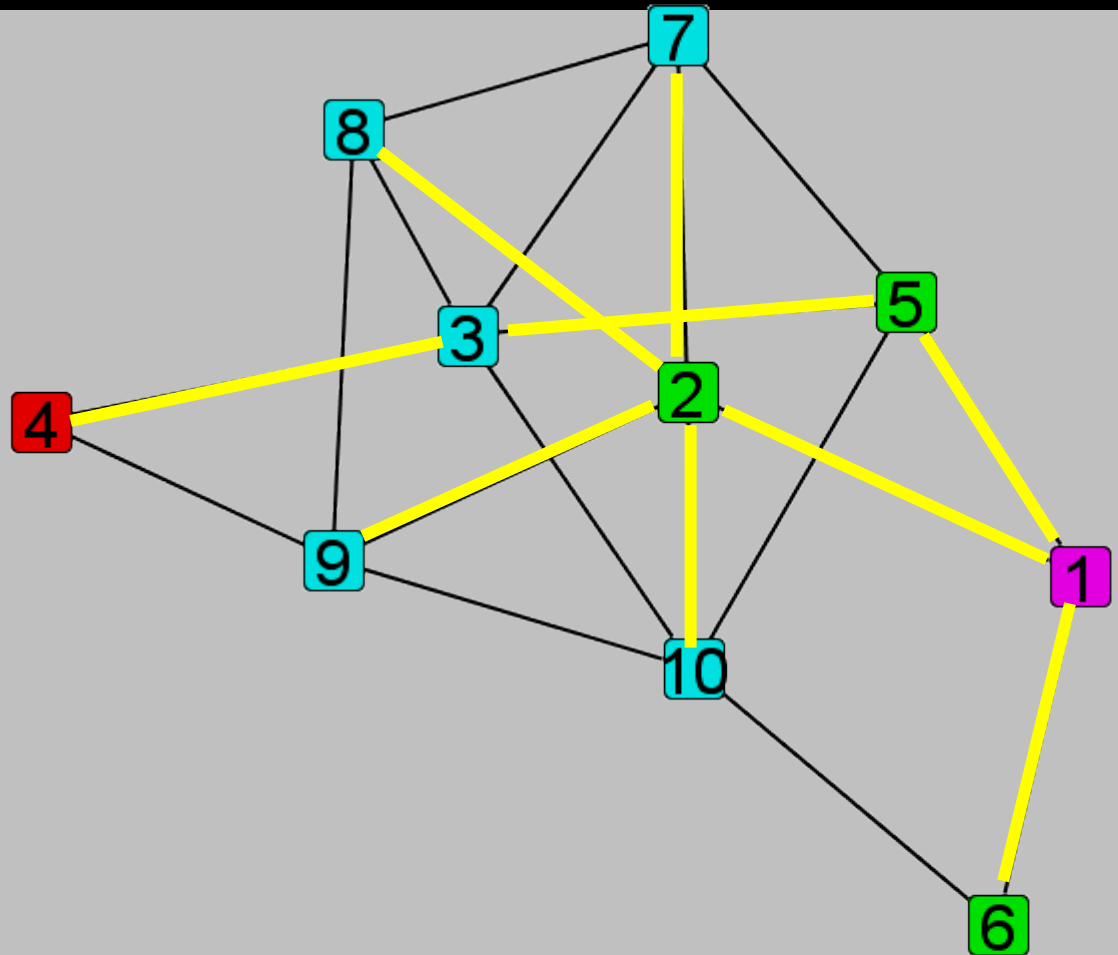
# Where are the paths?



- Distance 0
- Distance 1
- Distance 2
- Distance 3

# We computed paths while we were computing distances

- Distance 0
- Distance 1
- Distance 2
- Distance 3



# Broadcast Search

- A methodical way to compute shortest paths from one origin to the rest of the graph

Better known as  
Breadth-First-Search

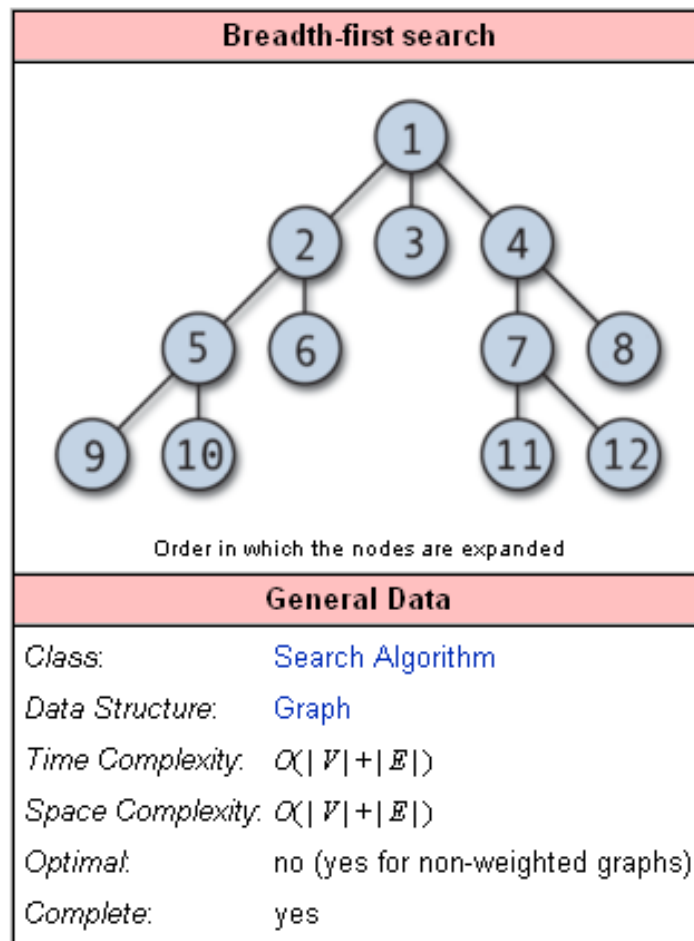
# Breadth-first search

From Wikipedia, the free encyclopedia

In [graph theory](#), **breadth-first search** (**BFS**) is a [graph search algorithm](#) that begins at the root [node](#) and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal.

## Contents [\[hide\]](#)

- How it works
- Algorithm (informal)
- Pseudocode
- Features
  - Space Complexity
  - Time Complexity
  - Completeness
  - Optimality
- Applications of BFS
  - Finding connected Components
  - Test for bipartiteness
- See also
- References
- External links



## How it works

BFS is an [uninformed search](#) method that aims to expand and examine all nodes of a [graph](#) systematically in search of a solution. In other words, it exhaustively searches the entire graph without considering the goal until it finds it. It does not use a [heuristic](#).

From the standpoint of the [algorithm](#), all child nodes obtained by expanding a node are added to a [FIFO queue](#). In typical implementations, nodes that have not yet been examined for their neighbors are placed in

## Graph search algorithms

### Search

- [A\\*](#)
- [Best-first search](#)
- [Bidirectional search](#)

# Breadth-first search

From Wikipedia, the free encyclopedia

In [graph theory](#), **breadth-first search** (**BFS**) is a [graph search algorithm](#) that begins at the root [node](#) and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal.

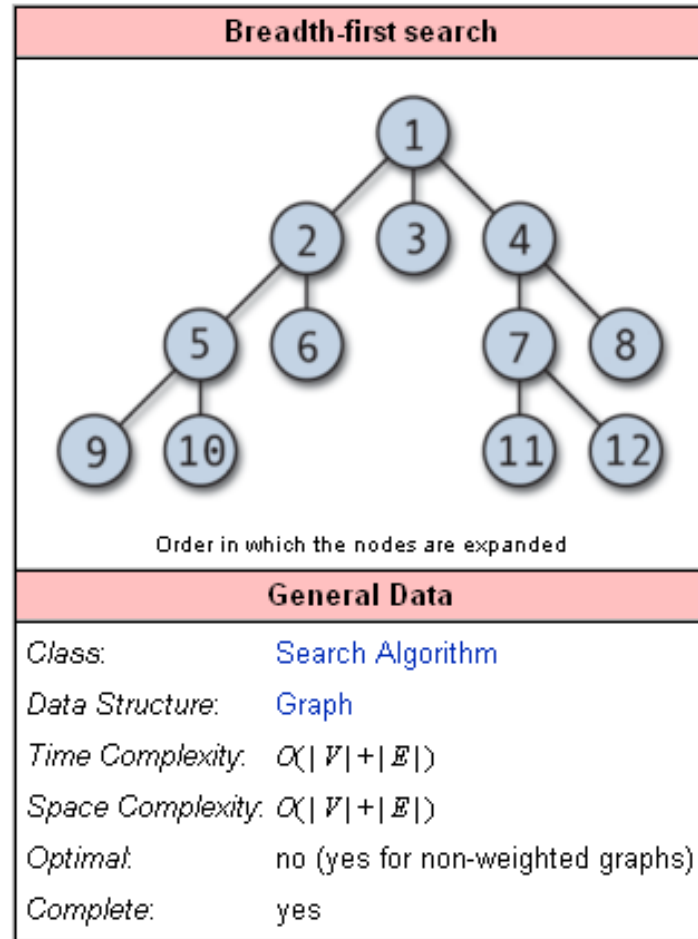
## Contents [\[hide\]](#)

- How it works
- Algorithm (informal)
- Pseudocode
- Features
  - Space Complexity
  - Time Complexity
  - Completeness
  - Optimality
- Applications of BFS
  - Finding connected Components
  - Test for bipartiteness
- See also
- References
- External links

Uninformed (e.g., Broadcast)

Vs

Informed (e.g., Directed)



## How it works

BFS is an [uninformed search](#) method that aims to expand and examine all nodes of a [graph](#) systematically in search of a solution. In other words, it exhaustively searches the entire graph without considering the goal until it finds it. It does not use a [heuristic](#).

From the standpoint of the [algorithm](#), all child nodes obtained by expanding a node are added to a [FIFO queue](#). In typical implementations, nodes that have not yet been examined for their neighbors are placed in

## Graph search algorithms

### Search

- A\*
- Best-first search
- Bidirectional search

# Breadth-first search

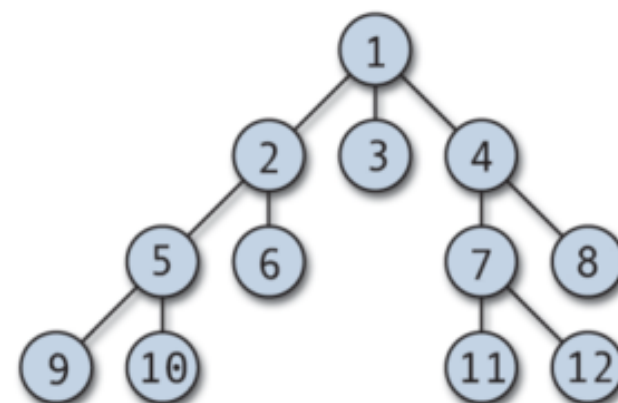
From Wikipedia, the free encyclopedia

In [graph theory](#), **breadth-first search** (**BFS**) is a [graph search algorithm](#) that begins at the root [node](#) and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal.

## Contents [\[hide\]](#)

- 1 How it works
- 2 Algorithm (informal)
- 3 Pseudocode
- 4 Features
  - 4.1 Space Complexity
  - 4.2 Time Complexity
  - 4.3 Completeness
  - 4.4 Optimality
- 5 Applications of BFS
  - 5.1 Finding connected Components
  - 5.2 Test for bipartiteness
- 6 See also
- 7 References
- 8 External links

**Heuristic**  
**Vs**  
**Algorithm**



Order in which the nodes are expanded

## General Data

<i>Class:</i>	Search Algorithm
<i>Data Structure:</i>	Graph
<i>Time Complexity:</i>	$O( V  +  E )$
<i>Space Complexity:</i>	$O( V  +  E )$
<i>Optimal:</i>	no (yes for non-weighted graphs)
<i>Complete:</i>	yes

[\[edit\]](#)

## How it works

BFS is an [uninformed search](#) method that aims to expand and examine all nodes of a [graph](#) systematically in search of a solution. In other words, it exhaustively searches the entire graph without considering the goal until it finds it. It does not use a [heuristic](#).

From the standpoint of the [algorithm](#), all child nodes obtained by expanding a node are added to a [FIFO queue](#). In typical implementations, nodes that have not yet been examined for their neighbors are placed in

## Graph search algorithms

### Search

- [A\\*](#)
- [Best-first search](#)
- [Bidirectional search](#)

# Breadth-first search

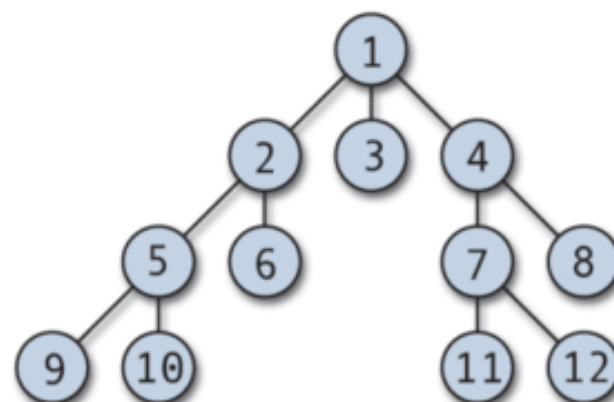
From Wikipedia, the free encyclopedia

In [graph theory](#), **breadth-first search** (**BFS**) is a [graph search algorithm](#) that begins at the root [node](#) and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal.

## Contents [\[hide\]](#)

- How it works
- Algorithm (informal)
- Pseudocode
- Features
  - Space Complexity
  - Time Complexity
  - Completeness
  - Optimality
- Applications of BFS
  - Finding connected Components
  - Test for bipartiteness
- See also
- References
- External links

**FIFO Queue**



Order in which the nodes are expanded

## General Data

<i>Class:</i>	Search Algorithm
<i>Data Structure:</i>	Graph
<i>Time Complexity:</i>	$O( V  +  E )$
<i>Space Complexity:</i>	$O( V  +  E )$
<i>Optimal:</i>	no (yes for non-weighted graphs)
<i>Complete:</i>	yes

## How it works

BFS is an [uninformed search](#) method that aims to expand and examine all nodes of a [graph](#) systematically in search of a solution. In other words, it exhaustively searches the entire graph without considering the goal until it finds it. It does not use a [heuristic](#).

From the standpoint of the [algorithm](#), all child nodes obtained by expanding a node are added to a [FIFO queue](#). In typical implementations, nodes that have not yet been examined for their neighbors are placed in

## Graph search algorithms

### Search

- A\*
- Best-first search
- Bidirectional search

# Breadth-first search

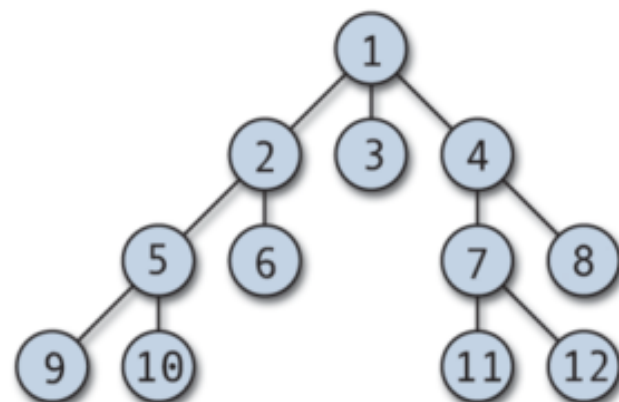
From Wikipedia, the free encyclopedia

In [graph theory](#), **breadth-first search** (**BFS**) is a [graph search algorithm](#) that begins at the root [node](#) and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal.

## Contents [\[hide\]](#)

- How it works
- Algorithm (informal)
- Pseudocode
- Features
  - Space Complexity
  - Time Complexity
  - Completeness
  - Optimality
- Applications of BFS
  - Finding connected Components
  - Test for bipartiteness
- See also
- References
- External links

**FIFO Queue**  
**“First In First Out”**



Order in which the nodes are expanded

## General Data

<i>Class:</i>	Search Algorithm
<i>Data Structure:</i>	Graph
<i>Time Complexity:</i>	$O( V  +  E )$
<i>Space Complexity:</i>	$O( V  +  E )$
<i>Optimal:</i>	no (yes for non-weighted graphs)
<i>Complete:</i>	yes

## How it works

BFS is an [uninformed search](#) method that aims to expand and examine all nodes of a [graph](#) systematically in search of a solution. In other words, it exhaustively searches the entire graph without considering the goal until it finds it. It does not use a [heuristic](#).

From the standpoint of the [algorithm](#), all child nodes obtained by expanding a node are added to a [FIFO queue](#). In typical implementations, nodes that have not yet been examined for their neighbors are placed in

## Graph search algorithms

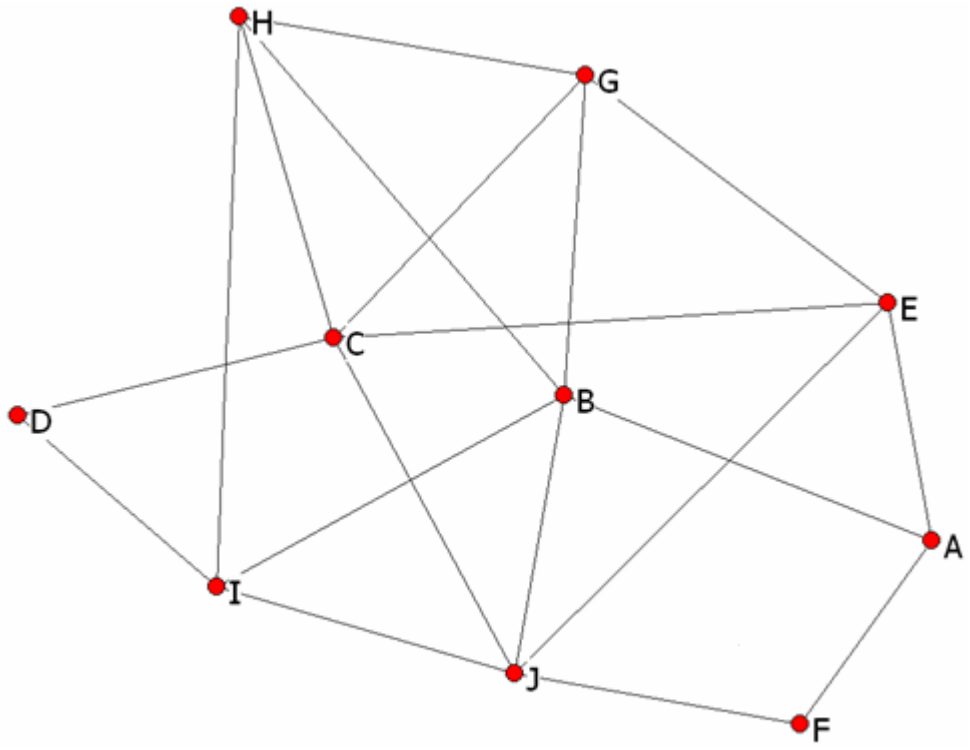
### Search

- A\*
- Best-first search
- Bidirectional search

# Broadcast Search Algorithm

The following algorithm computes shortest paths from a start node to every other node in a graph, using variables **distance[node]** and **predecessor[node]** to store results.

1. Put the start node (or *root* node) in the queue. Initialize  $\text{distance}[\text{root}] = 0$ .
2. If the queue is not empty, take the node from the **beginning** of the queue and call that node  $x$   
For each **unexamined** neighbor of  $x$ , call that node  $y$ 
  - Set  $\text{distance}[y] = \text{distance}[x] + 1$
  - Set  $\text{predecessor}[y] = x$
  - Add  $y$  to the **end** of the queue.
3. If the queue is empty, every node on the graph has been examined. Done.
4. Repeat from step 2.



Pred	Dist		A	B	C	D	E	F	G	H	I	J
		A	0	1	0	0	1	1	0	0	0	0
		B	1	0	0	0	0	0	1	1	1	1
		C	0	0	0	1	1	0	1	1	0	1
		D	0	0	1	0	0	0	0	0	1	0
		E	1	0	1	0	0	0	1	0	0	1
		F	1	0	0	0	0	0	0	0	0	1
		G	0	1	1	0	1	0	0	1	0	0
		H	0	1	1	0	0	0	1	0	1	0
		I	0	1	0	1	0	0	0	1	0	1
		J	0	1	1	0	1	1	0	0	1	0