Algorithms

Lecture Topic: Representation of Graphs, BFS

Anxiao (Andrew) Jiang

Roadmap of this lecture:

1. Representation of Graphs.

    1.1 Define graphs.

    1.2 Represent graphs by "Adjacency List" and "Adjacency Matrix".

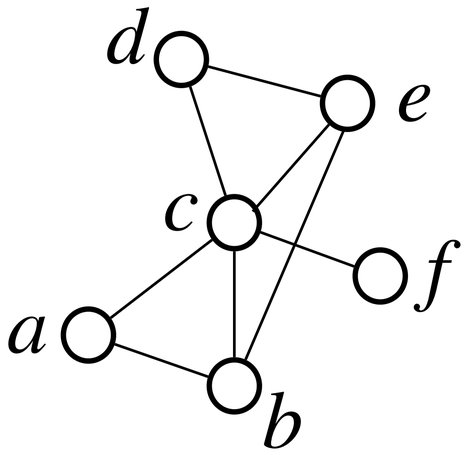
2. Breadth First Search (BFS).

Representation of Graphs

Graph

Node
(vertex)

Edge
(Undirected,
Directed)



$$G = (V, E)$$

$$V = \{a, b, c, d, e, f\}$$

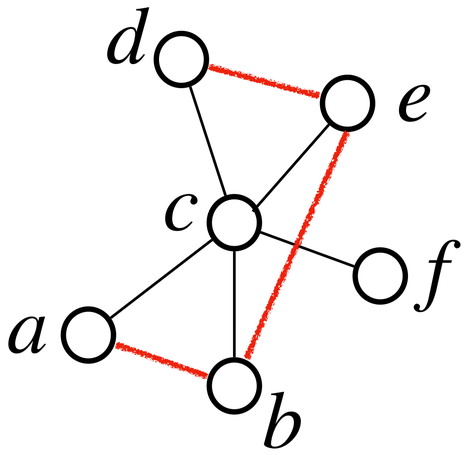
$$E = \{(a, b), (a, c), (b, c), \cdots, (c, f)\}$$

Representation of Graphs

Graph

Node
(vertex)

Edge
(Undirected,
Directed)



Path

$$G = (V, E)$$

$$V = \{a, b, c, d, e, f\}$$

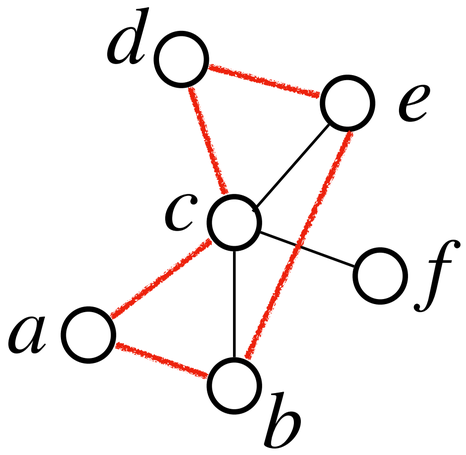
$$E = \{(a, b), (a, c), (b, c), \cdots, (c, f)\}$$

Representation of Graphs

Graph

Node
(vertex)

Edge
(Undirected,
Directed)



Cycle

$$G = (V, E)$$

$$V = \{a, b, c, d, e, f\}$$

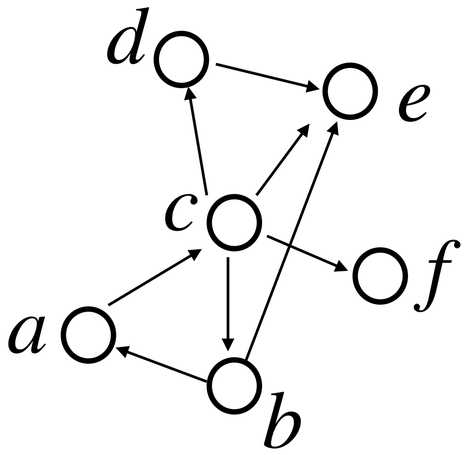
$$E = \{(a, b), (a, c), (b, c), \cdots, (c, f)\}$$

Representation of Graphs

Graph

Node
(vertex)

Edge
(Undirected,
Directed)



$$G = (V, E)$$

$$V = \{a, b, c, d, e, f\}$$

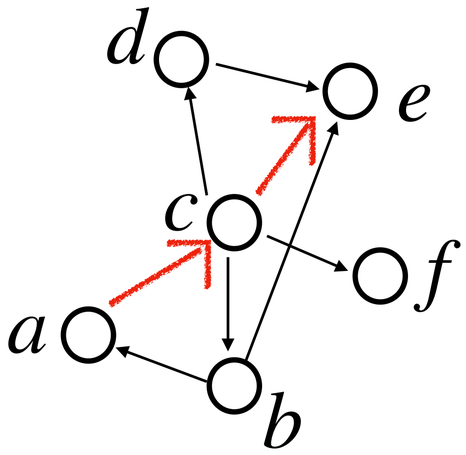
$$E = \{(a, c), (b, a), (b, e), \cdots, (c, f)\}$$

Representation of Graphs

Graph

Node
(vertex)

Edge
(Undirected,
Directed)



Path

$$G = (V, E)$$

$$V = \{a, b, c, d, e, f\}$$

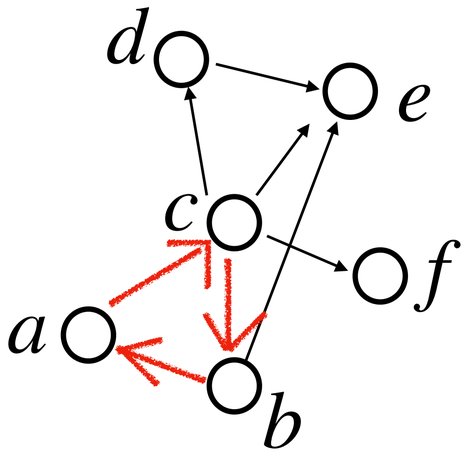
$$E = \{(a, c), (b, a), (b, e), \cdots, (c, f)\}$$

Representation of Graphs

Graph

Node
(vertex)

Edge
(Undirected,
Directed)



Cycle

$$G = (V, E)$$

$$V = \{a, b, c, d, e, f\}$$

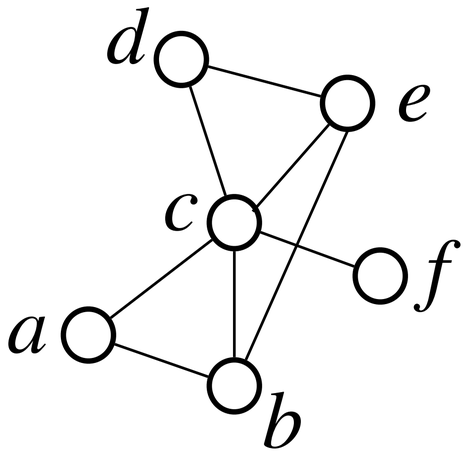
$$E = \{(a, c), (b, a), (b, e), \cdots, (c, f)\}$$

Representation of Graphs

Graph

Node
(vertex)

Edge
(Undirected,
Directed)

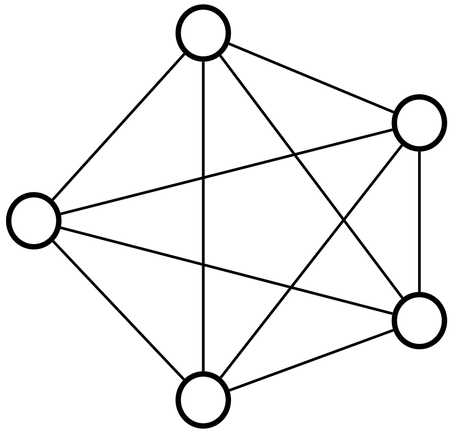


$$G = (V, E)$$

$$V = \{a, b, c, d, e, f\}$$

$$E = \{(a, b), (a, c), (b, c), \cdots, (c, f)\}$$

Complete Graph:

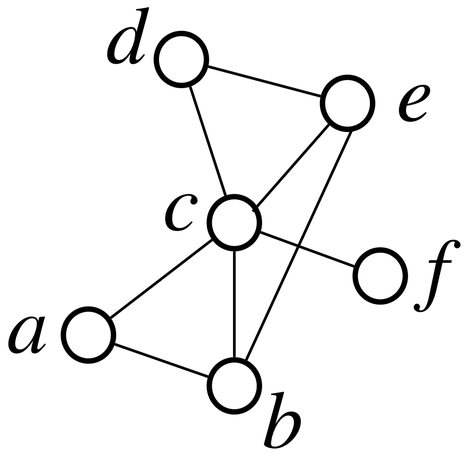


Representation of Graphs

Graph

Node
(vertex)

Edge
(Undirected,
Directed)

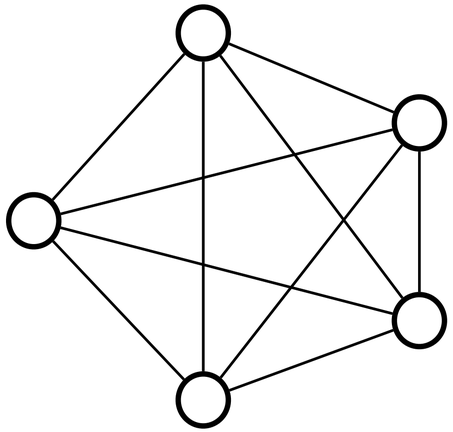


$$G = (V, E)$$

$$V = \{a, b, c, d, e, f\}$$

$$E = \{(a, b), (a, c), (b, c), \cdots, (c, f)\}$$

Complete Graph:



Tree: connected acyclic graph.

Number of edges = number of nodes - 1

Quiz question:

1. What is the difference between directed and undirected graphs?

Roadmap of this lecture:

1. Representation of Graphs.

    1.1 Define graphs.

    1.2 Represent graphs by "Adjacency List" and "Adjacency Matrix".

2. Breadth First Search (BFS).

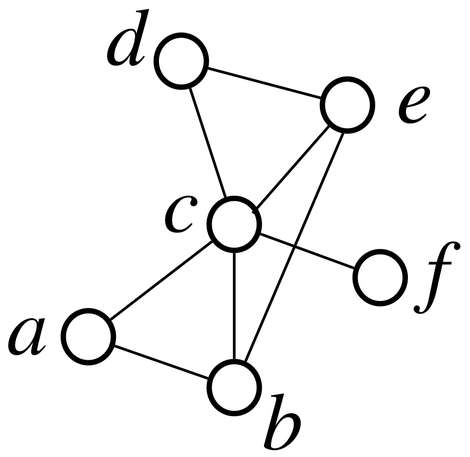
Representation of Graphs

Adjacency List

$a:\ b,\ c$
$b:\ a,\ c,\ e$
$c:\ a,\ b,\ d,\ e,\ f$
$d:\ c,\ e$
$e:\ b,\ c,\ d$
$f:\ c$

Size: $O(|V| + |E|)$

$O(V + E)$



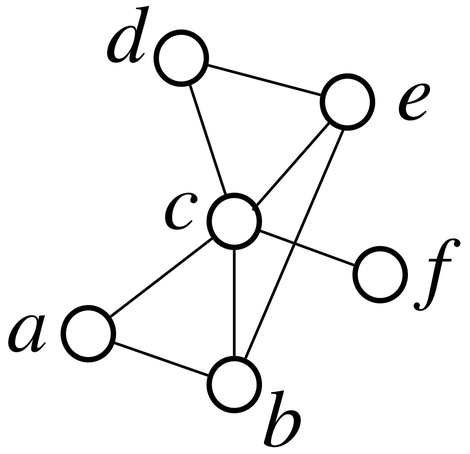
Representation of Graphs

Adjacency List

Size: $O(|V| + |E|)$

$O(V + E)$

$a: b, c$
$b: a, c, e$
$c: a, b, d, e, f$
$d: c, e$
$e: b, c, d$
$f: c$



Adjacency Matrix

Size: $O(|V|^2)$

$O(V^2)$

	a	b	c	d	e	f
a	0	1	1	0	0	0
b	1	0	1	0	1	0
c	1	1	0	1	1	1
d	0	0	1	0	1	0
e	0	1	1	1	0	0
f	0	0	1	0	0	0

Quiz questions:

1. What circumstances make it convenient to use “Adjacency List” to represent a graph?

2. What circumstances make it convenient to use “Adjacency Matrix” to represent a graph?

Roadmap of this lecture:

1. Representation of Graphs.

    1.1 Define graphs.

    1.2 Represent graphs by "Adjacency List" and "Adjacency Matrix".

2. Breadth First Search (BFS).

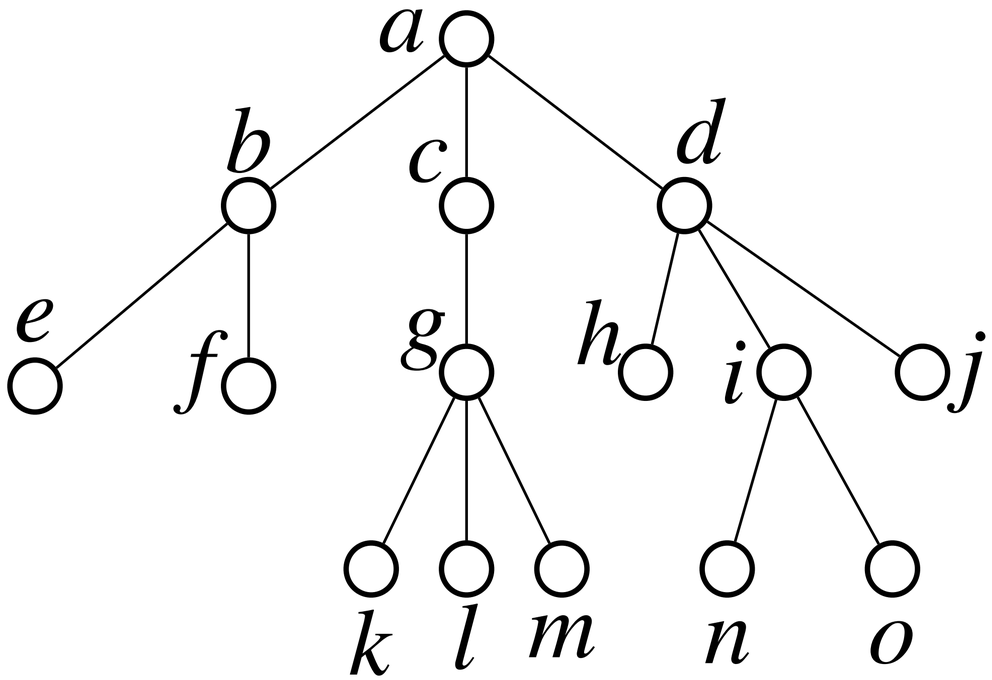
Breadth First Search (BFS)

Basic idea:

Starting at a node,
We first check its 1-hop neighbors,
then check its 2-hop neighbors (neighbors of 1-hop neighbors),
then check its 3-hop neighbors (neighbors of 2-hop neighbors),
then check its 4-hop neighbors (neighbors of 3-hop neighbors),
……,
until all reachable nodes are checked.

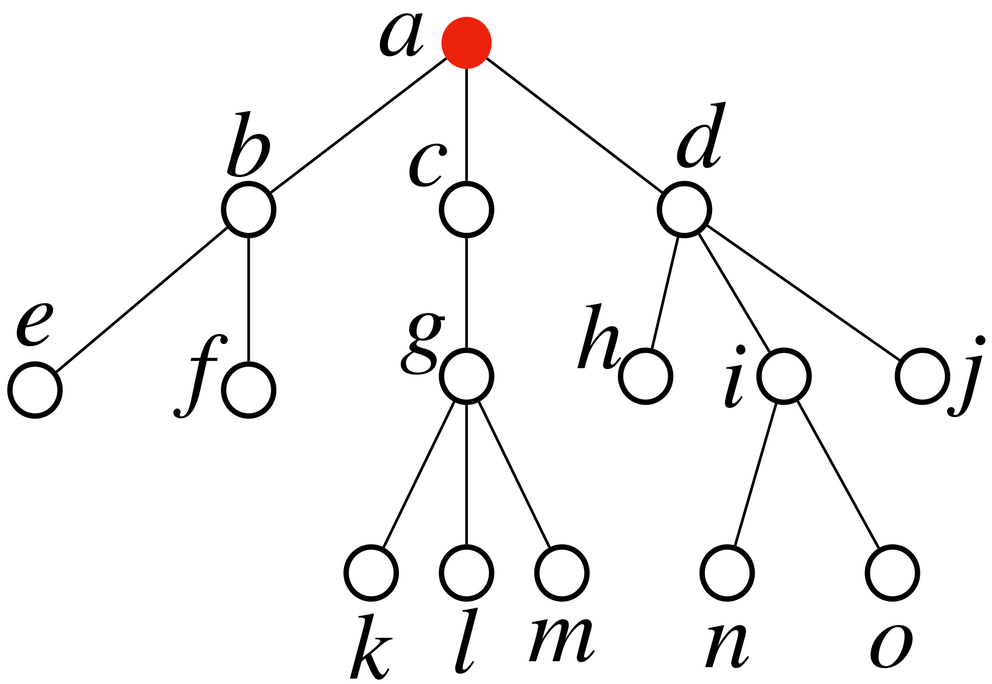
Breadth First Search (BFS)

Example of BFS on a tree:



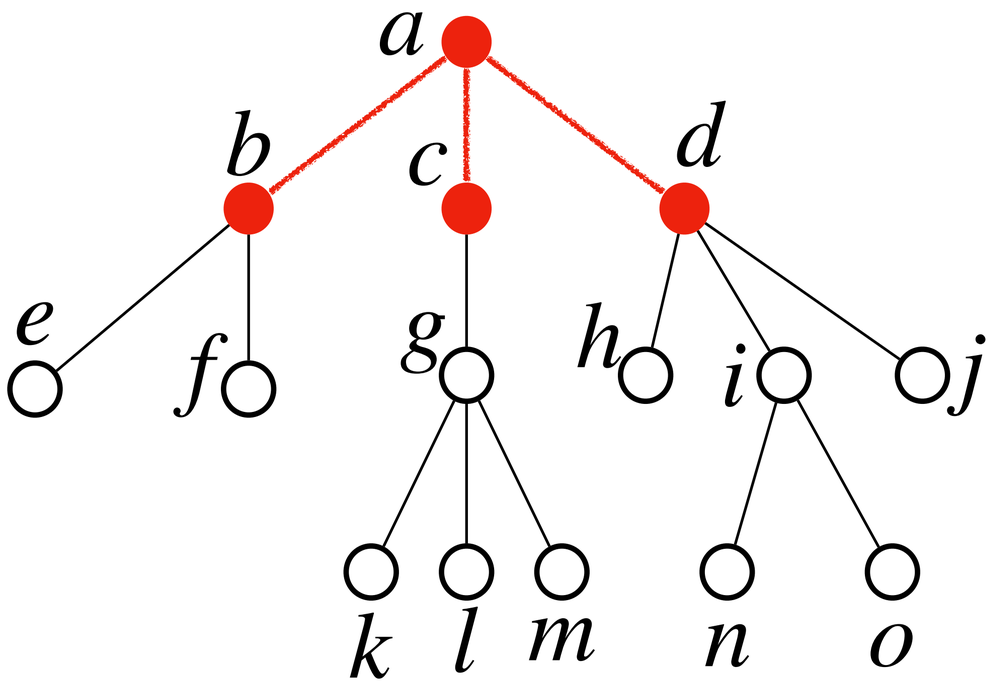
Breadth First Search (BFS)

Example of BFS on a tree:



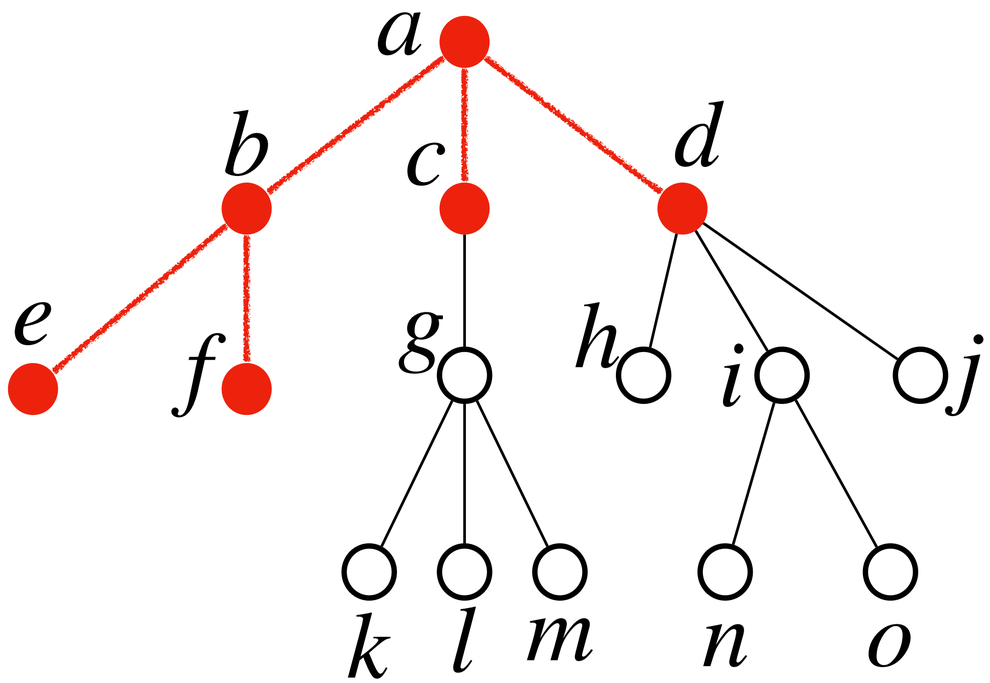
Breadth First Search (BFS)

Example of BFS on a tree:



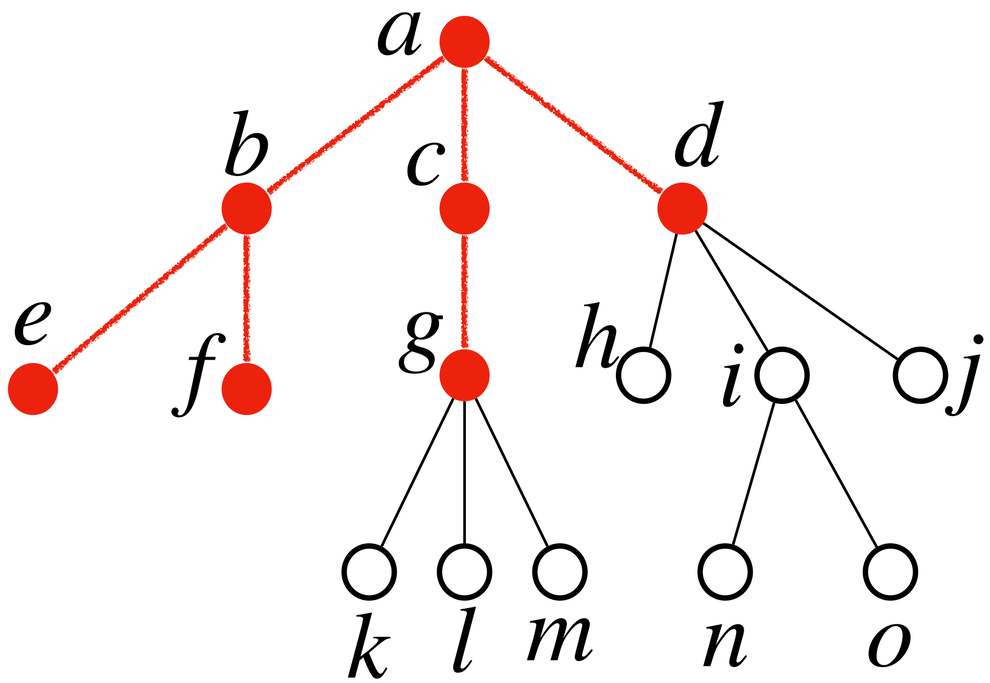
Breadth First Search (BFS)

Example of BFS on a tree:



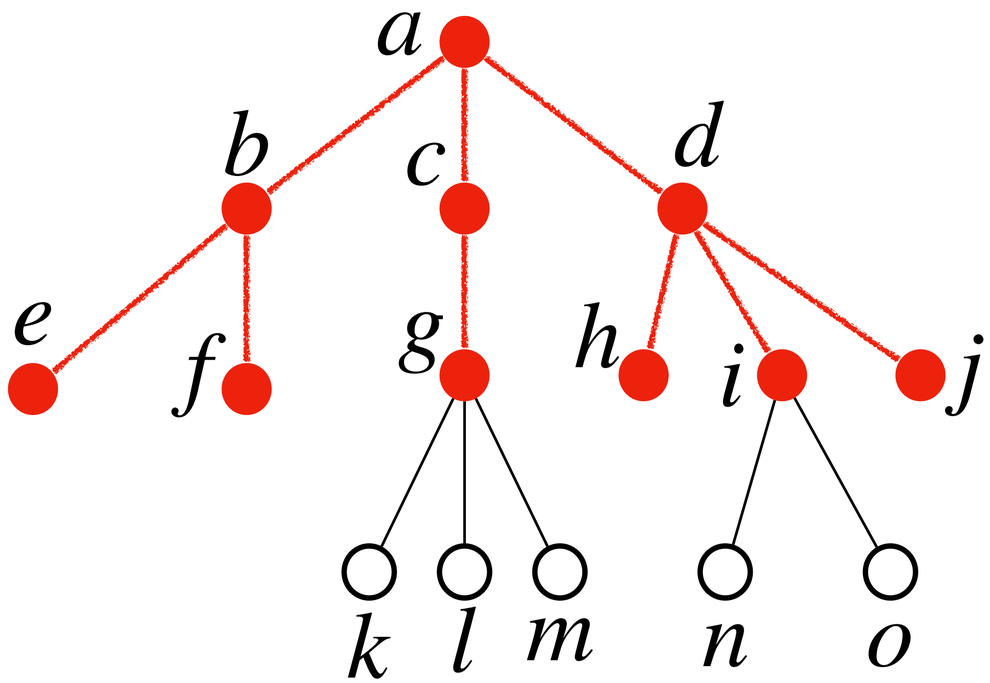
Breadth First Search (BFS)

Example of BFS on a tree:



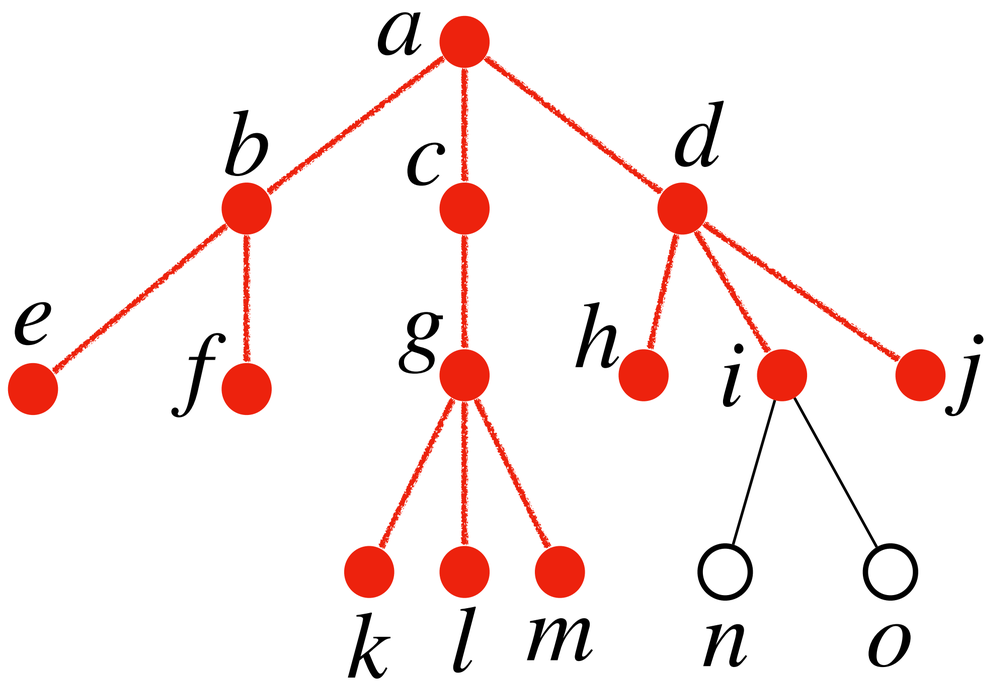
Breadth First Search (BFS)

Example of BFS on a tree:



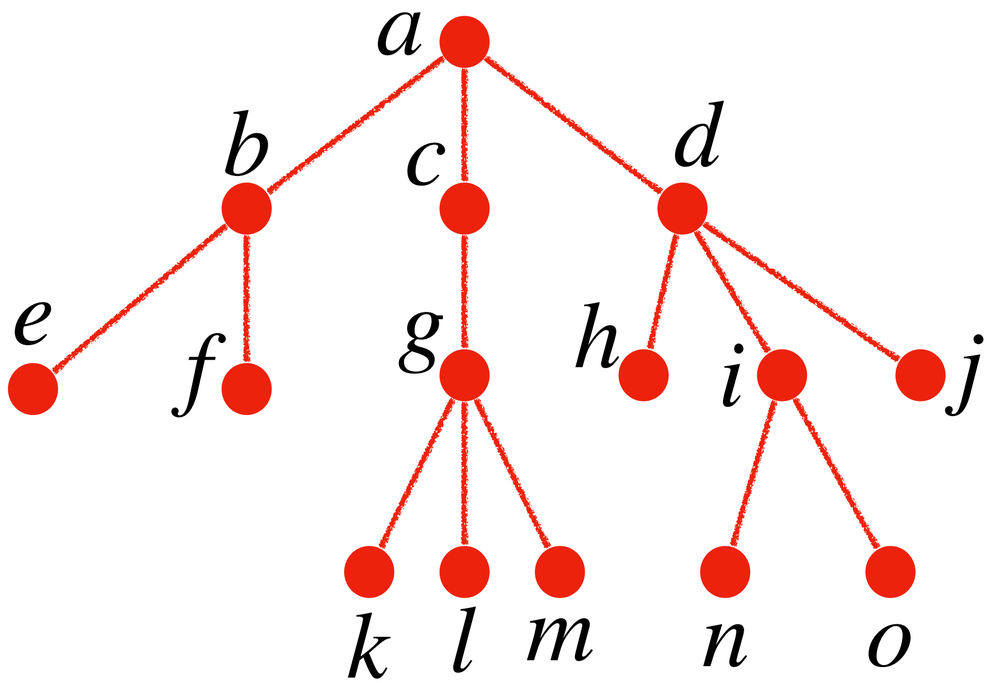
Breadth First Search (BFS)

Example of BFS on a tree:

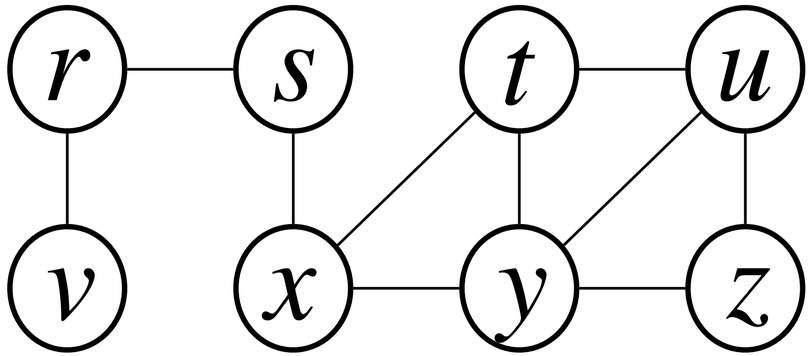


Breadth First Search (BFS)

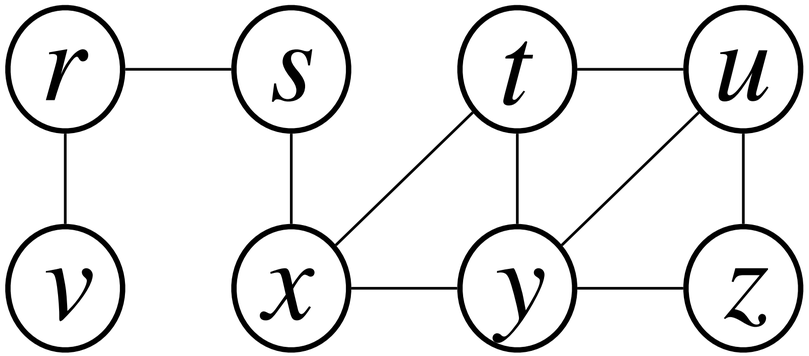
Example of BFS on a tree:



Breadth First Search (BFS)



Breadth First Search (BFS)

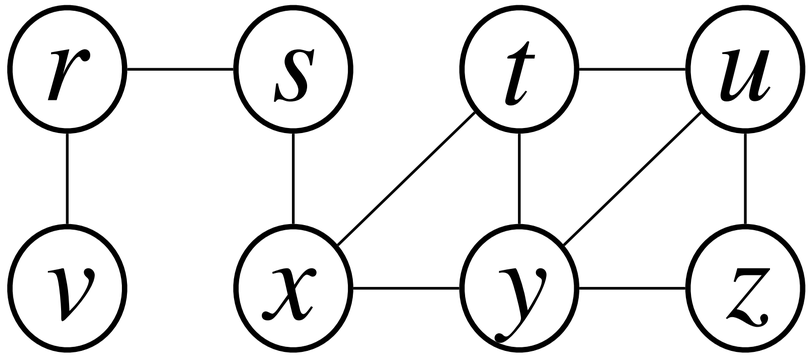


Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

Finished nodes: black

Breadth First Search (BFS)



Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

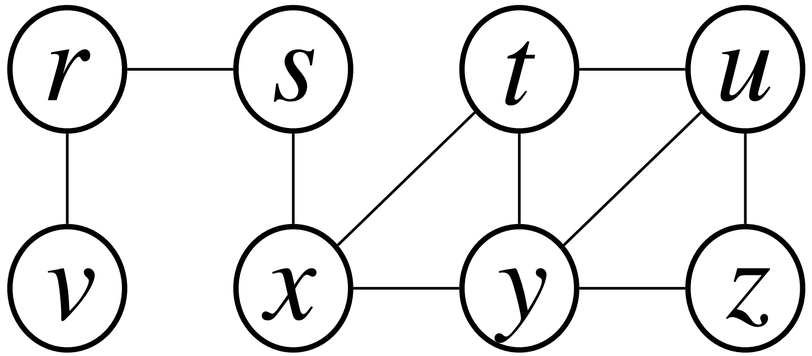
Finished nodes: black

For each node v：

1) distance from root to v： $d(v)$

2) the node that discovered v： $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

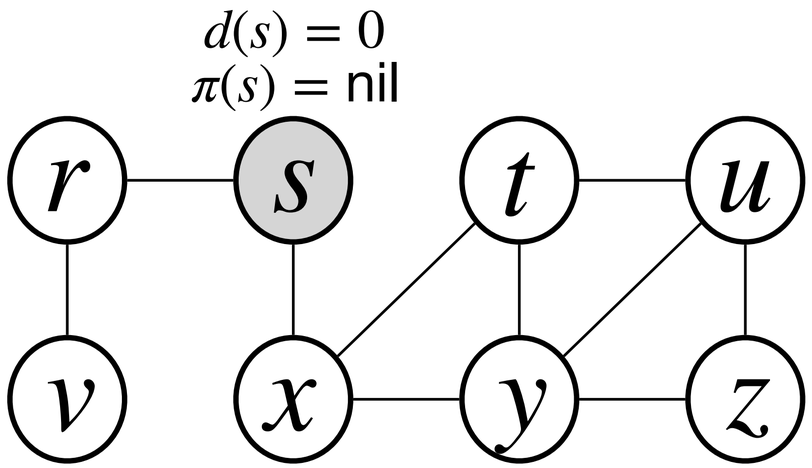
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

Finished nodes: black

For each node v :

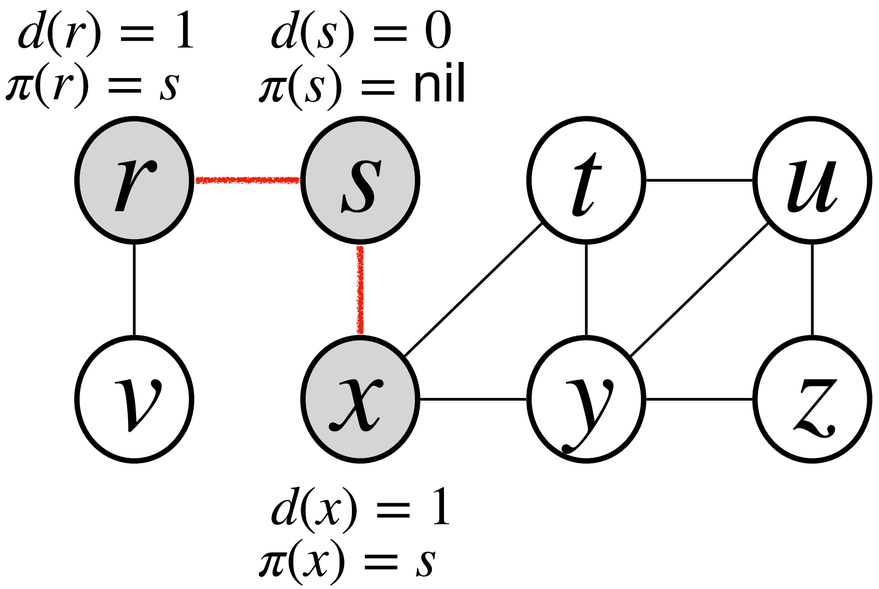
1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Queue (for remembering the discovered nodes):

s

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

s r x

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

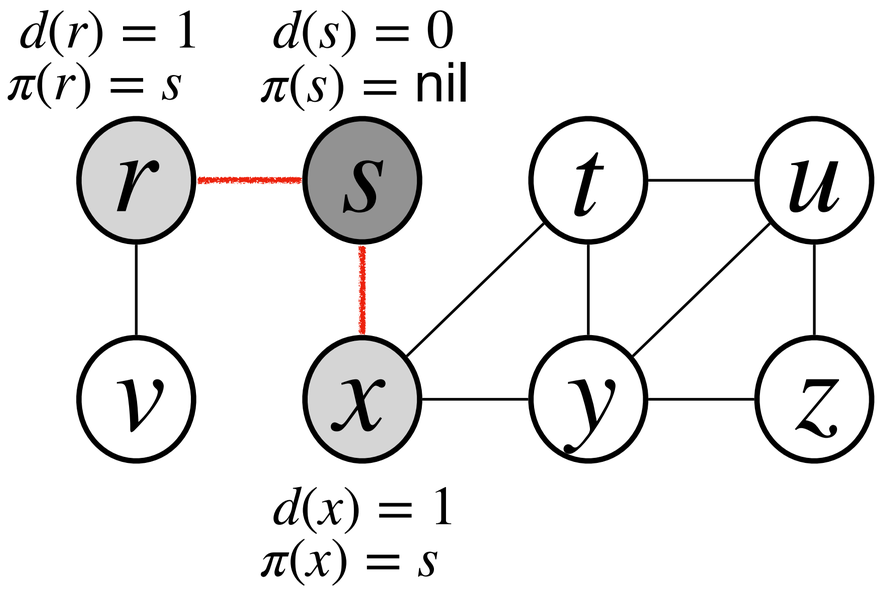
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

r x

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

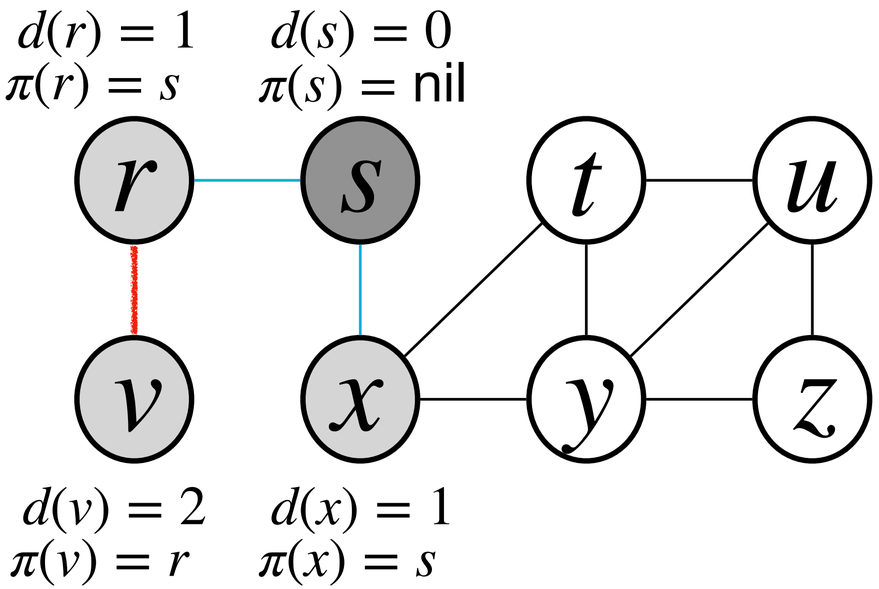
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

r x v

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

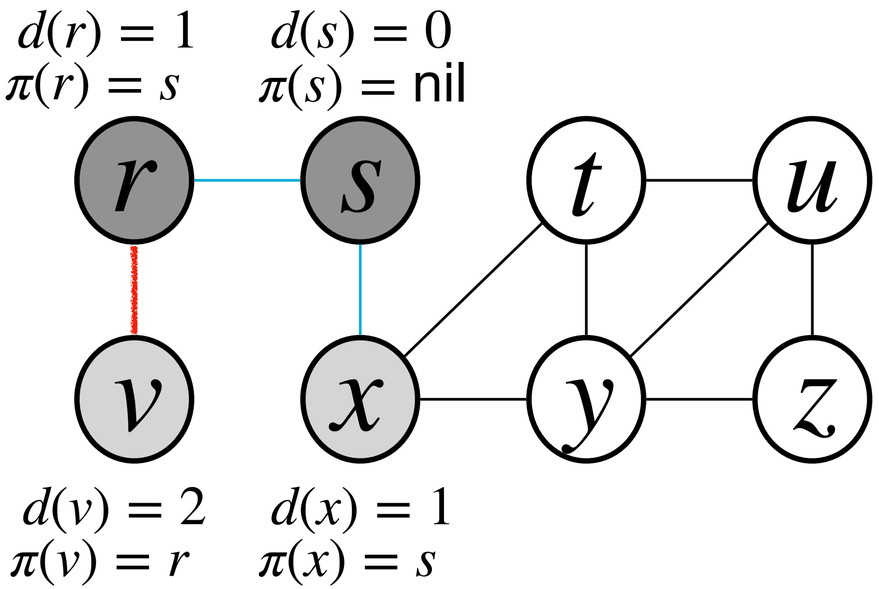
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

x v

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

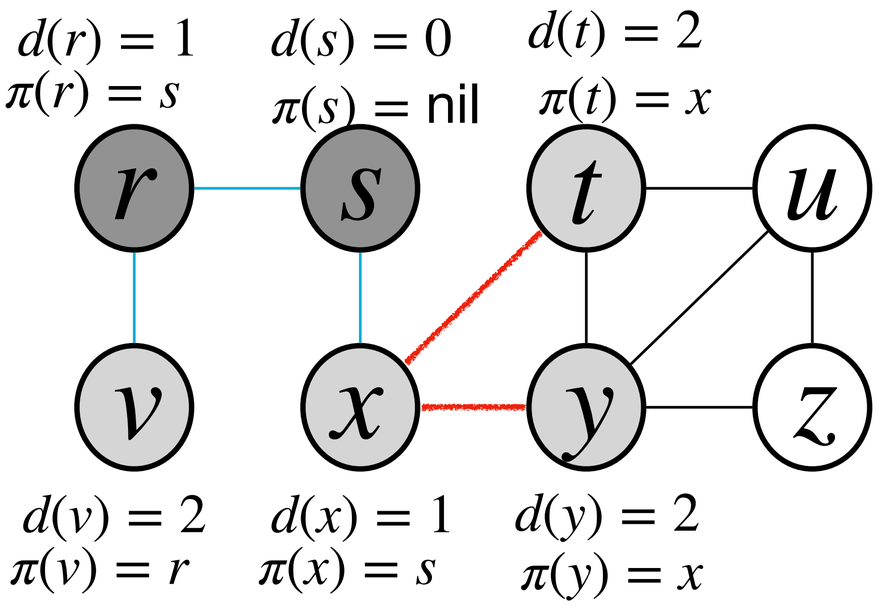
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

x v t y

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

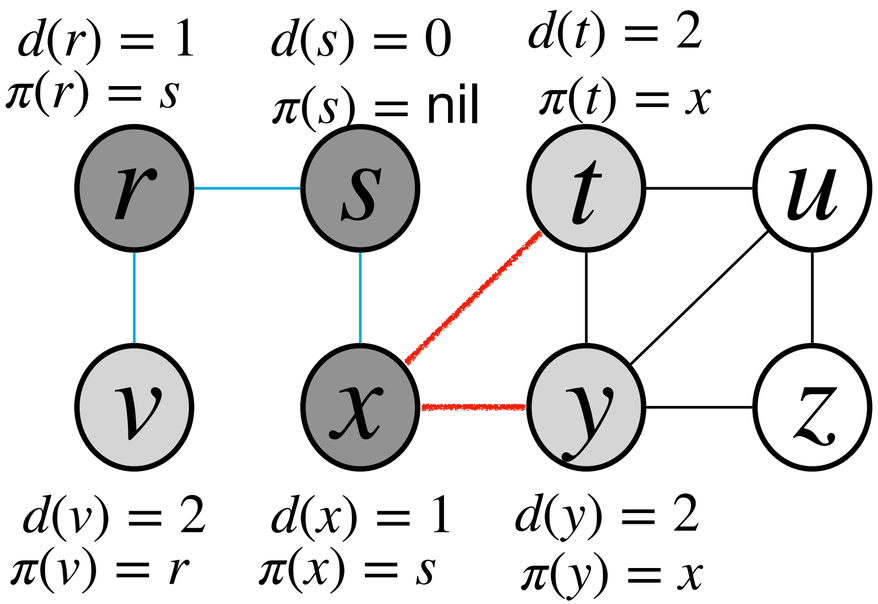
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

v t y

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

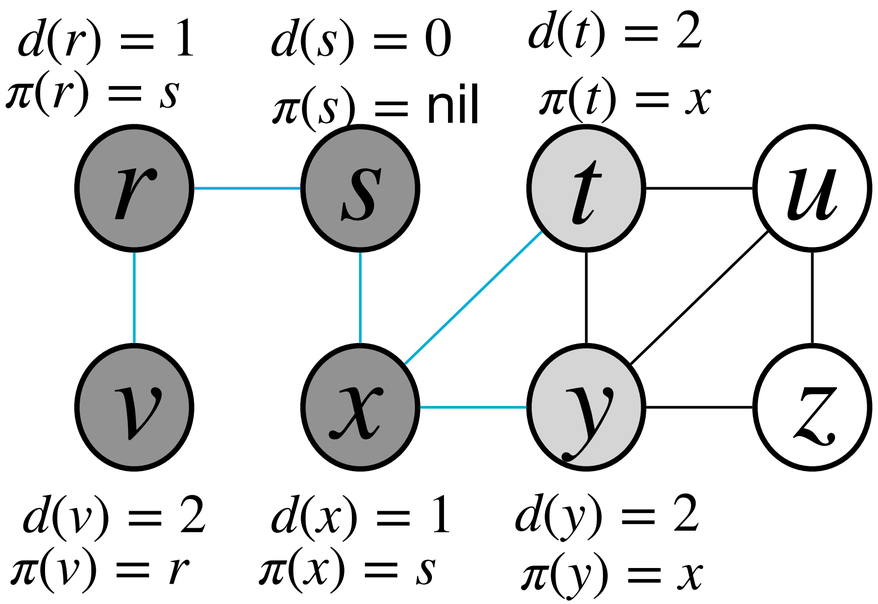
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

t y

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

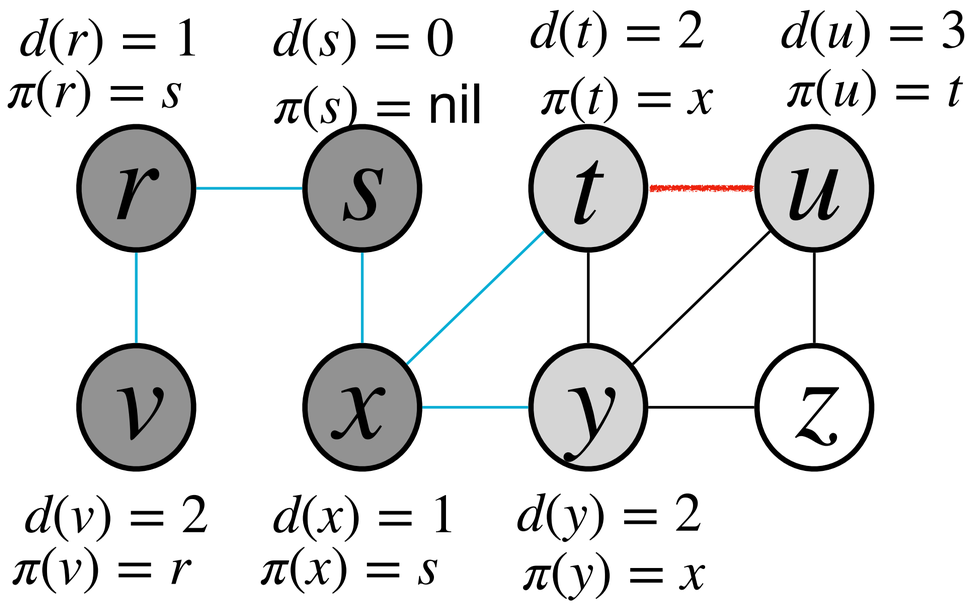
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

t y u

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

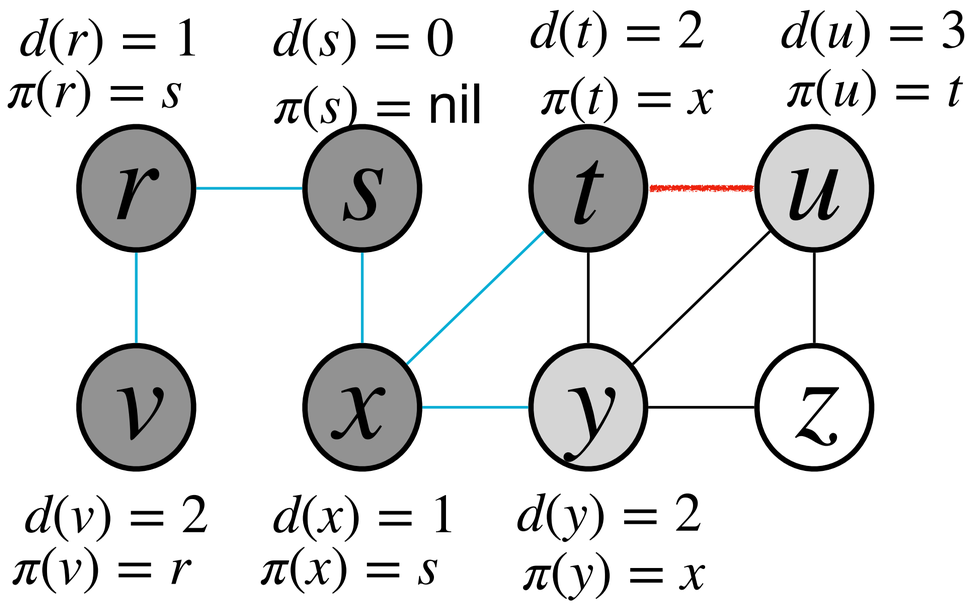
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

y u

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

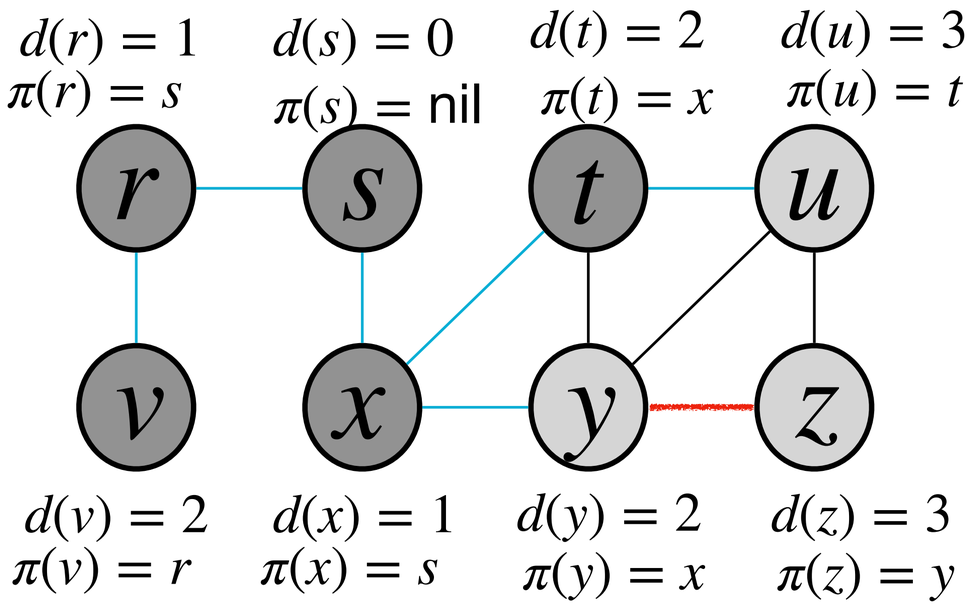
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

y u z

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

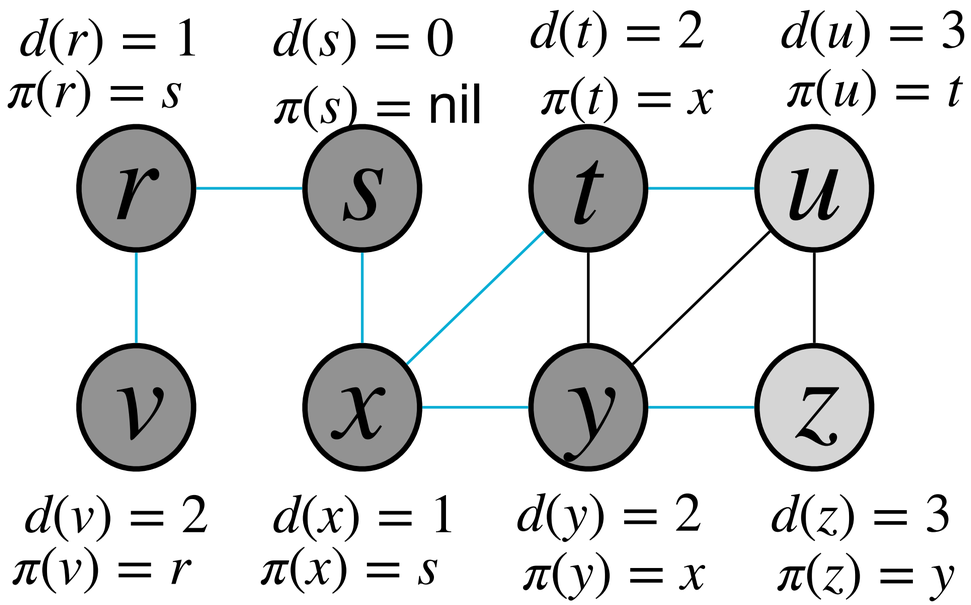
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

u z

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

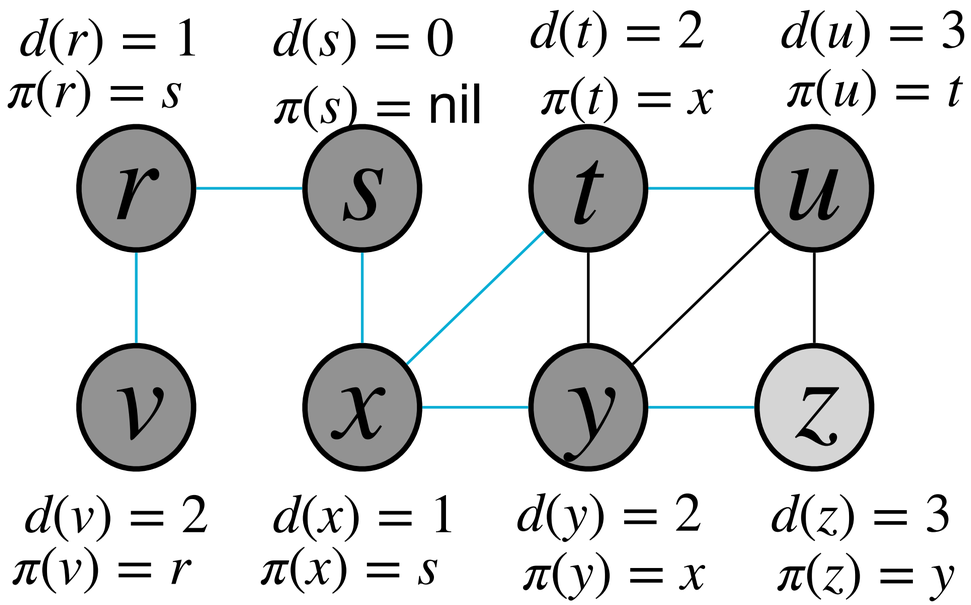
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

z

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

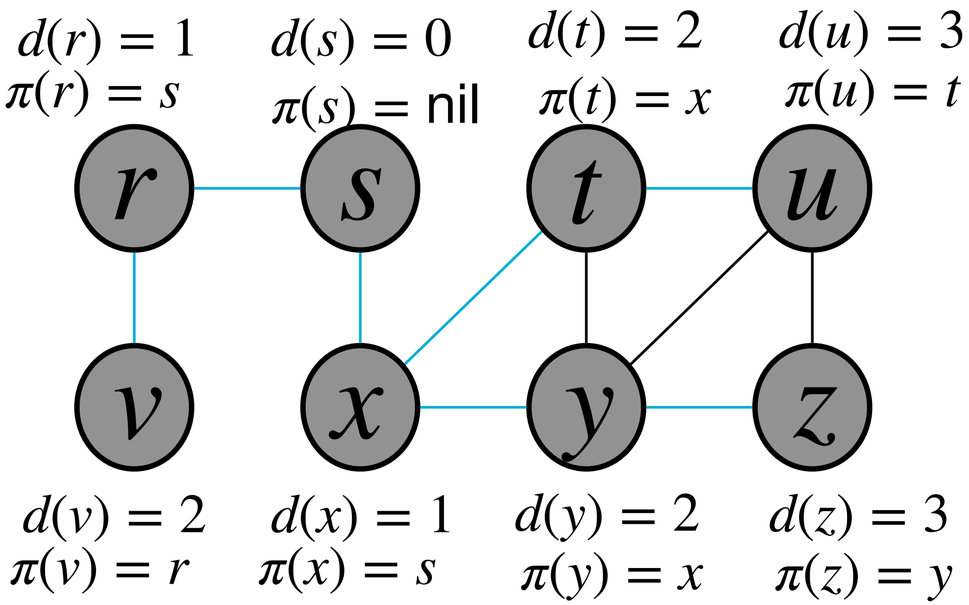
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



Queue (for remembering the discovered nodes):

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

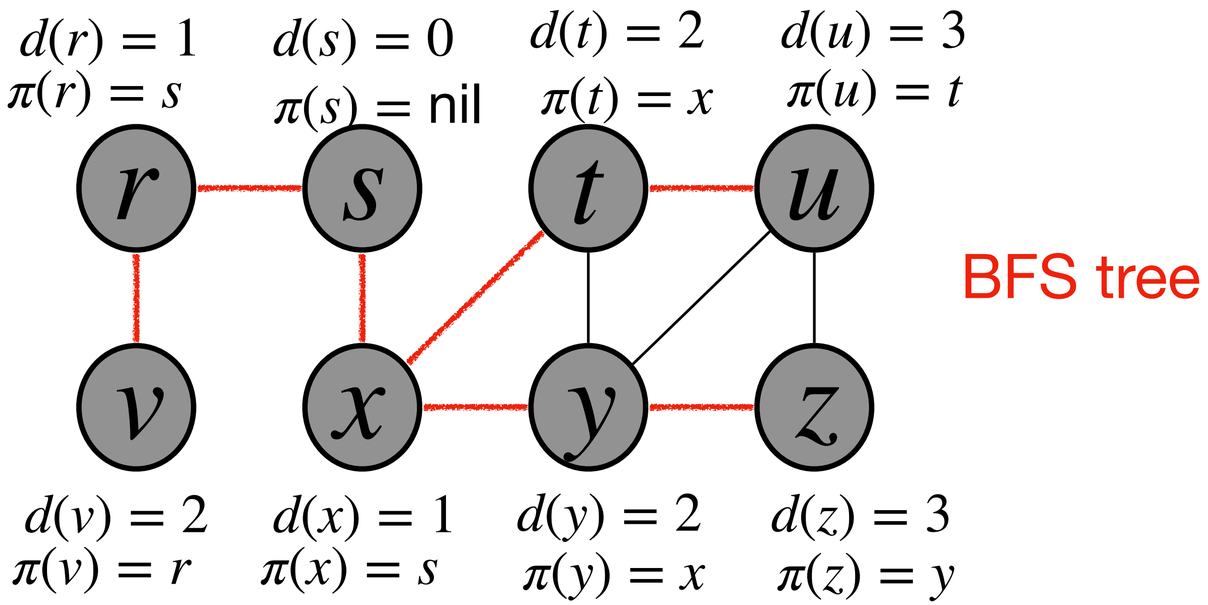
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



BFS tree

Queue (for remembering the discovered nodes):

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

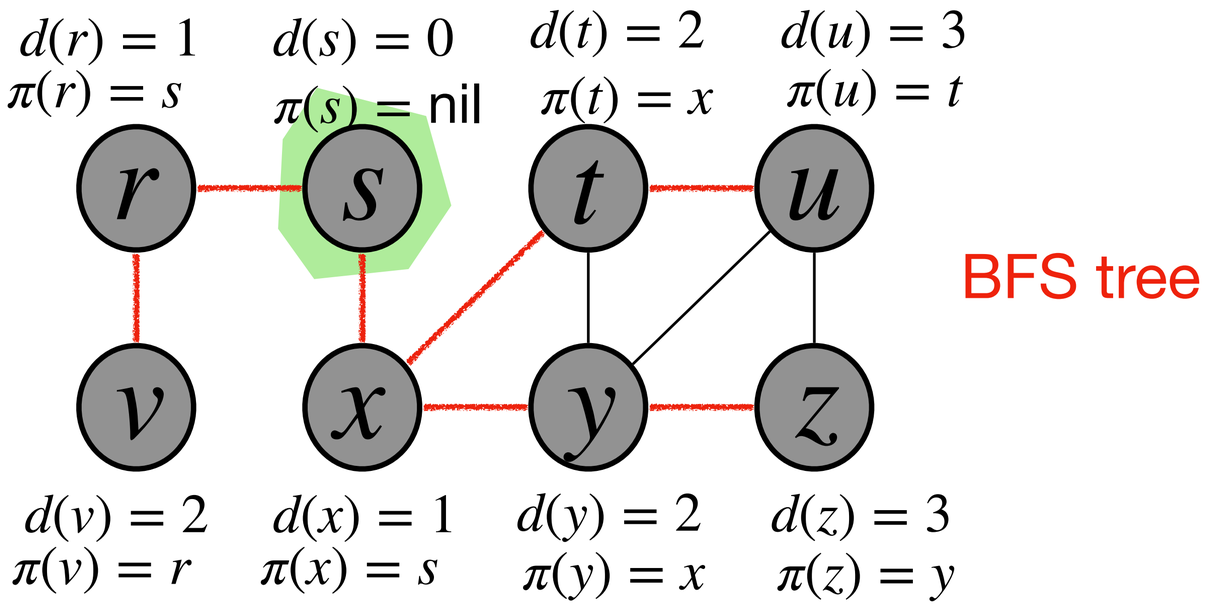
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



BFS tree

Queue (for remembering the discovered nodes):

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

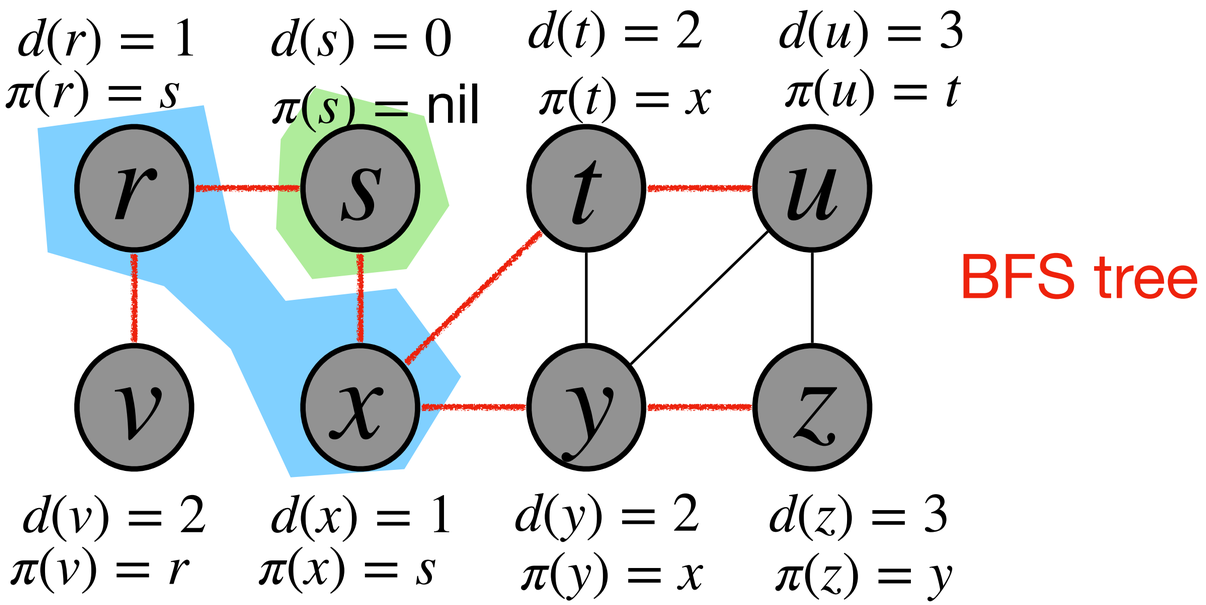
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



BFS tree

Queue (for remembering the discovered nodes):

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

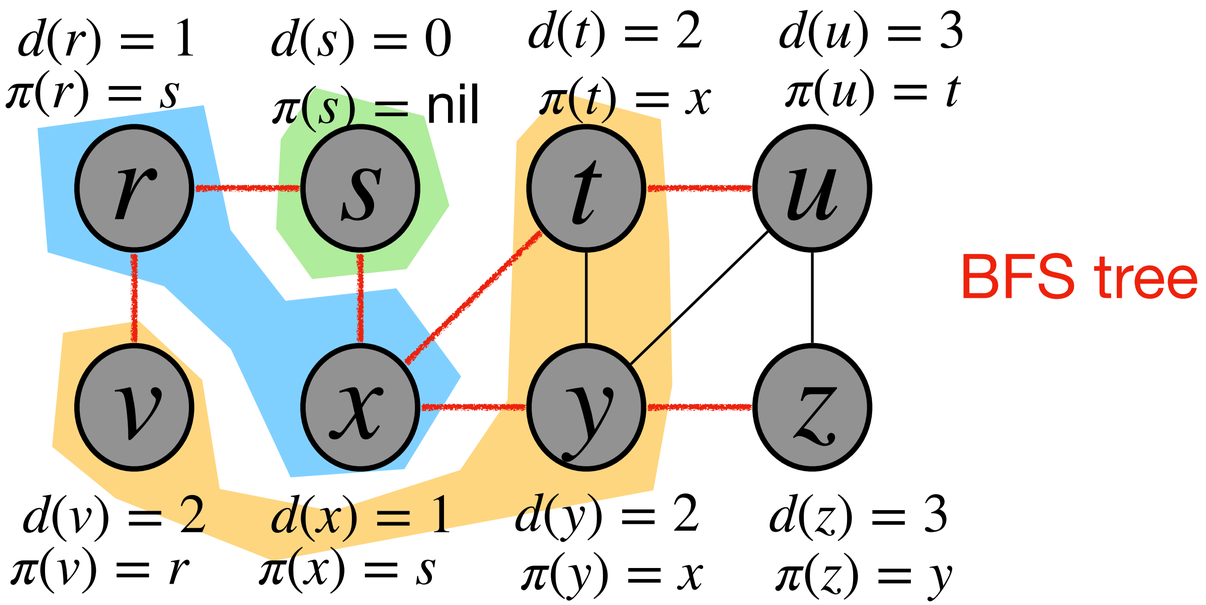
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



BFS tree

Queue (for remembering the discovered nodes):

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

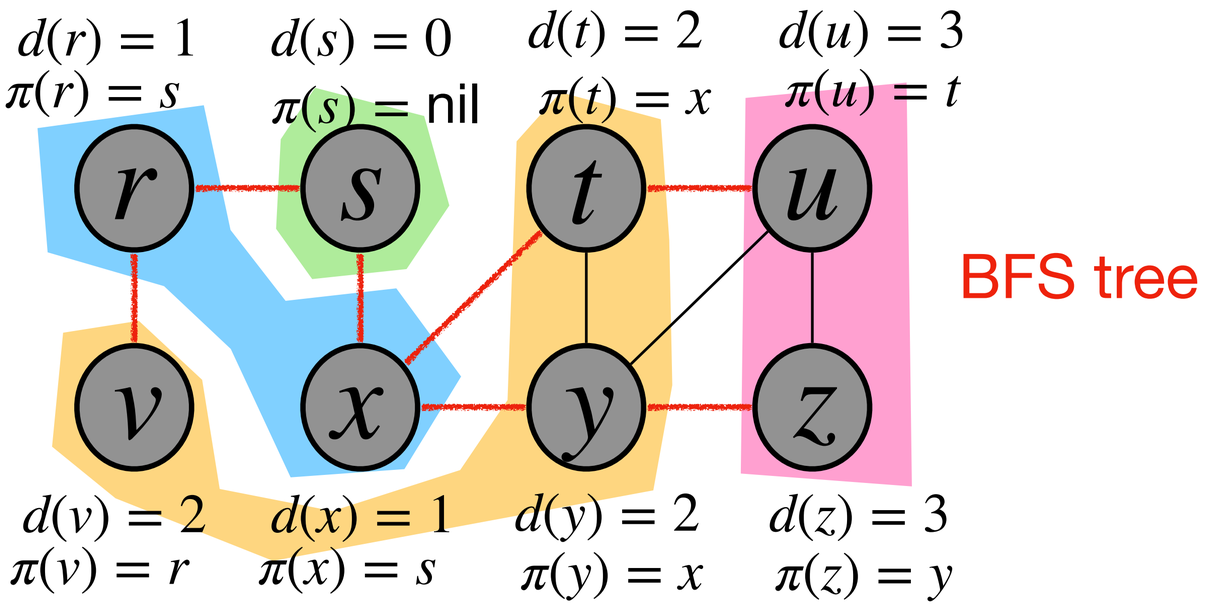
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



BFS tree

Queue (for remembering the discovered nodes):

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

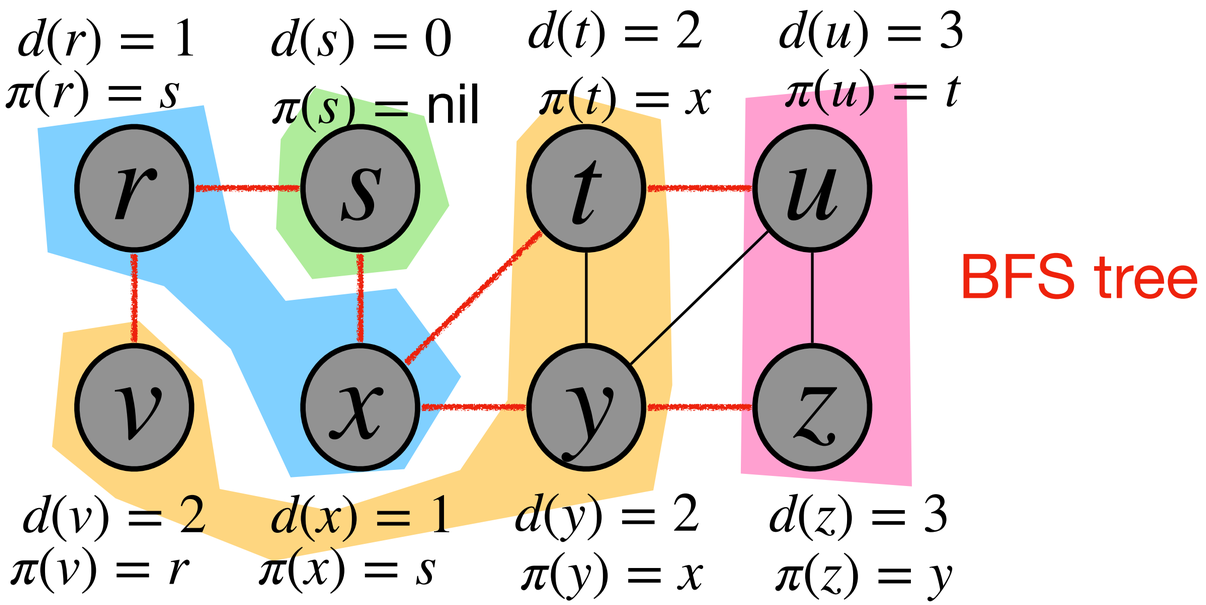
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



BFS tree

Queue (for remembering the discovered nodes):

$s,\ r,\ x,\ v,\ t,\ y,\ u,\ z$

0 1 2 3

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

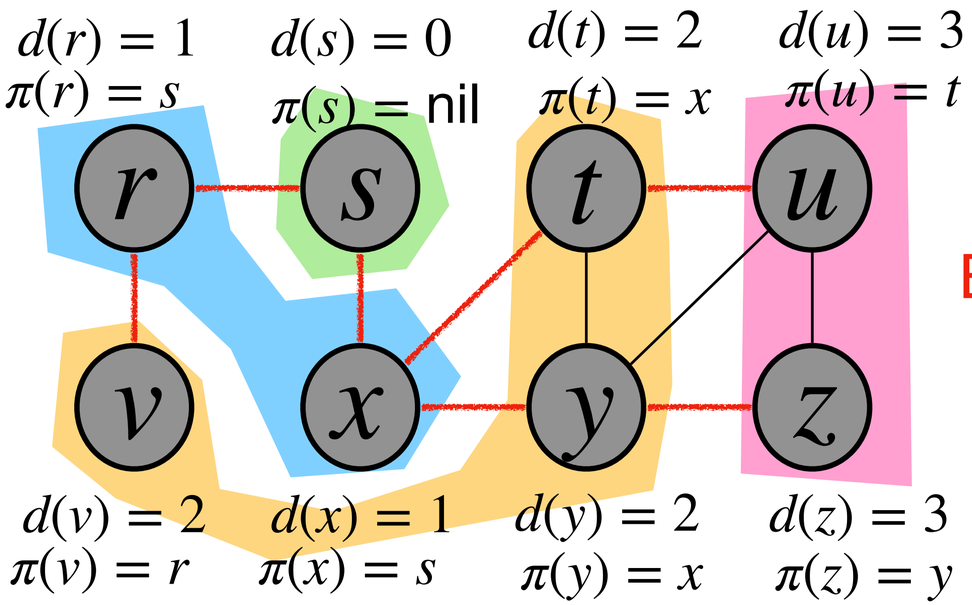
Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Breadth First Search (BFS)



BFS tree

Undiscovered nodes: white

Discovered (but unfinished) nodes: gray

Finished nodes: black

For each node v :

1) distance from root to v : $d(v)$

2) the node that discovered v : $\pi(v)$

Queue (for remembering the discovered nodes):

$s,\ r,\ x,\ v,\ t,\ y,\ u,\ z$

0 1 2 3

Time complexity: $O(V + E)$

Quiz questions:

1. How does BFS find the minimum distance from the root to all vertices (in terms of edge hops)?

2. Why is the time complexity of BFS O(V+E)?