Applications of BFS and DFS

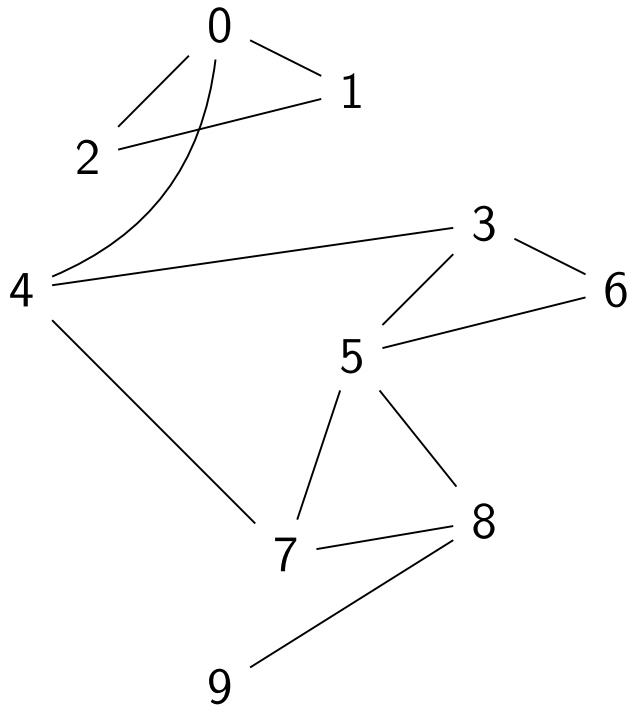
Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Mathematics for Data Science 1
Week 10

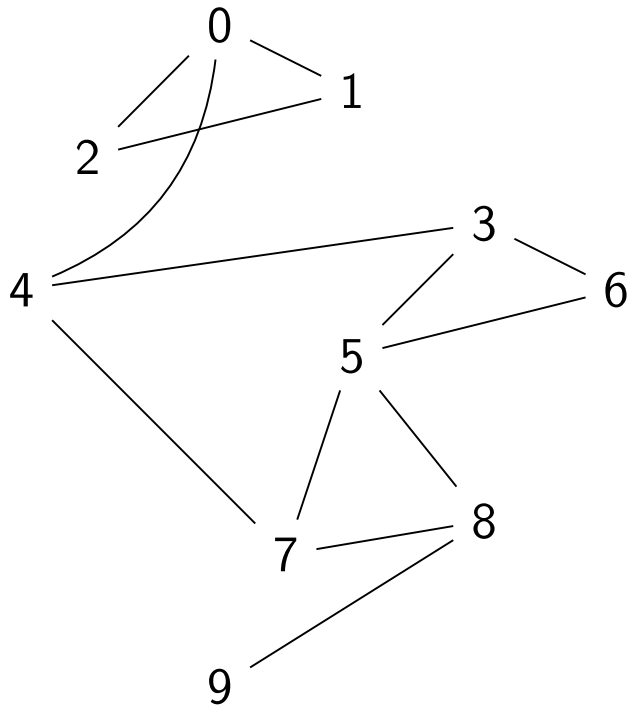
BFS and DFS

- BFS and DFS systematically compute reachability in graphs



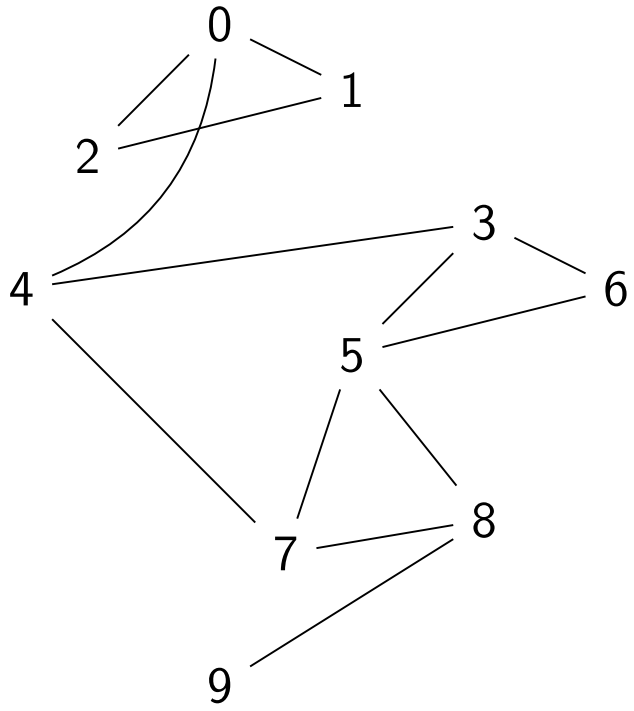
BFS and DFS

- BFS and DFS systematically compute reachability in graphs
- BFS works level by level
  - Discovers shortest paths in terms of number of edges



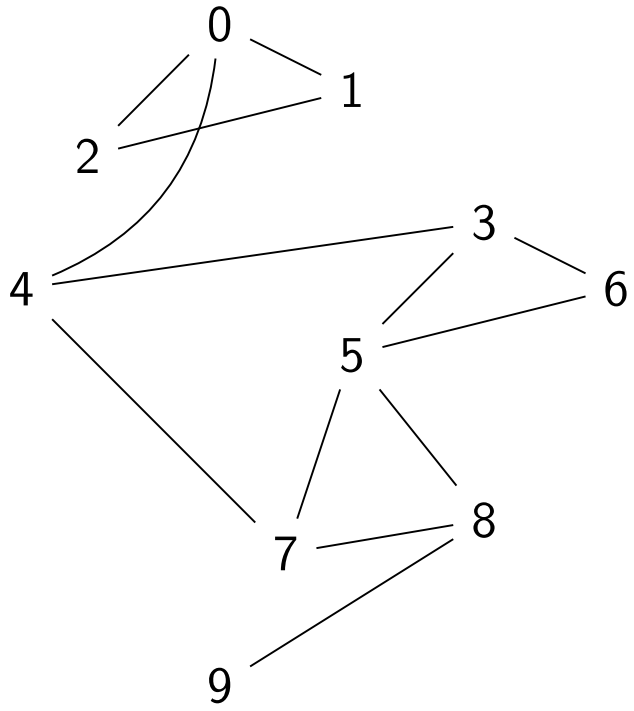
BFS and DFS

- BFS and DFS systematically compute reachability in graphs
- BFS works level by level
  - Discovers shortest paths in terms of number of edges
- DFS explores a vertex as soon as it is visited neighbours
  - Suspend a vertex while exploring its neighbours
  - DFS numbering describes the order in which vertices are explored



BFS and DFS

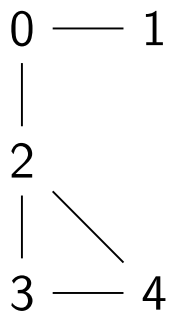
- BFS and DFS systematically compute reachability in graphs
- BFS works level by level
  - Discovers shortest paths in terms of number of edges
- DFS explores a vertex as soon as it is visited neighbours
  - Suspend a vertex while exploring its neighbours
  - DFS numbering describes the order in which vertices are explored
- Beyond reachability, what can we find out about a graph using BFS/DFS?



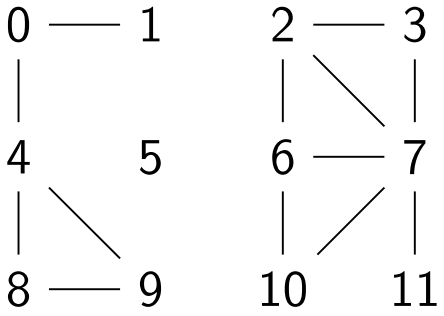
Connectivity

- An undirected graph is connected if every vertex is reachable from every other vertex

Connected Graph



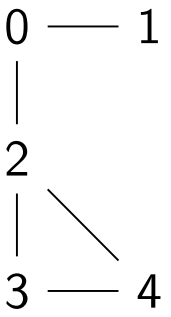
Disconnected Graph



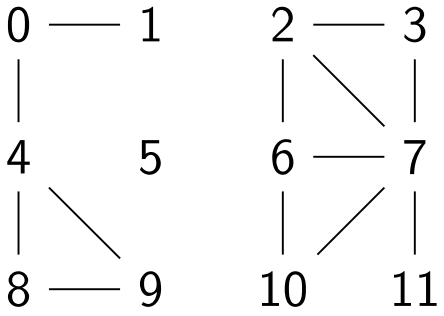
Connectivity

- An undirected graph is connected if every vertex is reachable from every other vertex
- In a disconnected graph, we can identify the connected components

Connected Graph



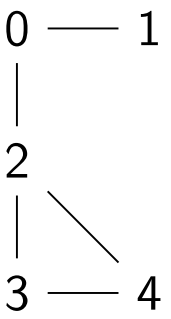
Disconnected Graph



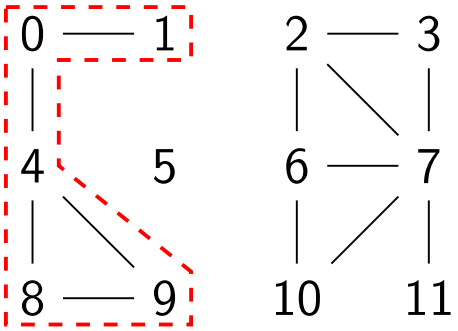
Connectivity

- An undirected graph is connected if every vertex is reachable from every other vertex
- In a disconnected graph, we can identify the connected components
  - Maximal subsets of vertices that are connected

Connected Graph



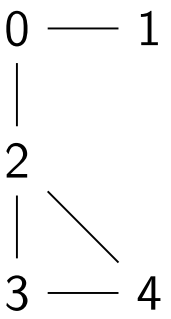
Disconnected Graph



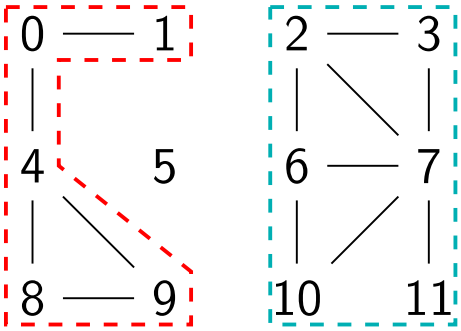
Connectivity

- An undirected graph is connected if every vertex is reachable from every other vertex
- In a disconnected graph, we can identify the connected components
  - Maximal subsets of vertices that are connected

Connected Graph



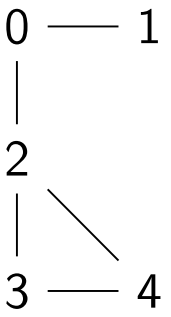
Disconnected Graph



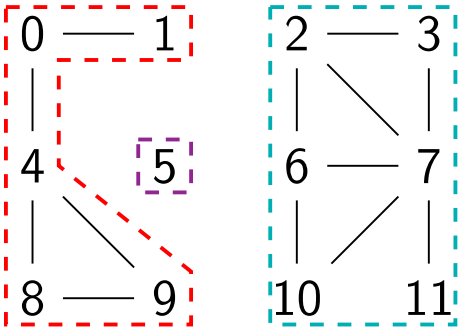
Connectivity

- An undirected graph is connected if every vertex is reachable from every other vertex
- In a disconnected graph, we can identify the connected components
  - Maximal subsets of vertices that are connected
  - Isolated vertices are trivial components

Connected Graph



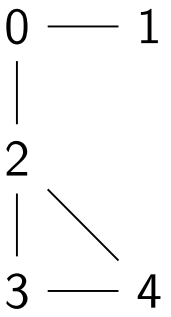
Disconnected Graph



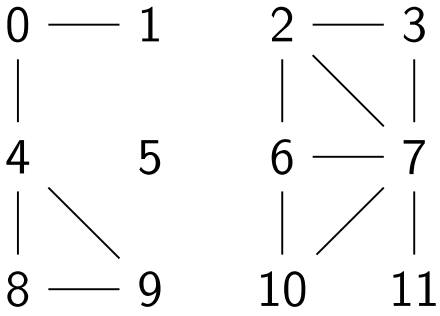
Identifying connected components

- Assign each vertex a component number

Connected Graph



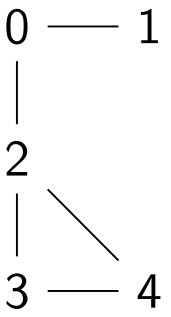
Disconnected Graph



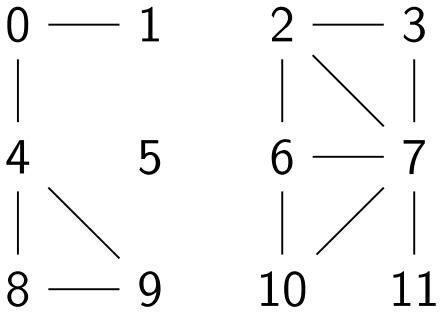
Identifying connected components

- Assign each vertex a component number
- Start BFS/DFS from vertex 0

Connected Graph



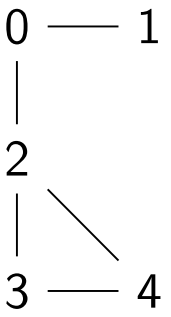
Disconnected Graph



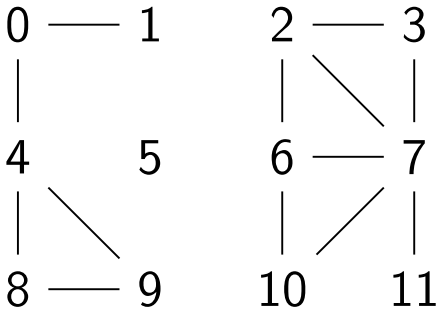
Identifying connected components

- Assign each vertex a component number
- Start BFS/DFS from vertex 0
  - Initialize component number to 0

Connected Graph



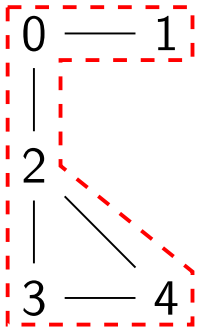
Disconnected Graph



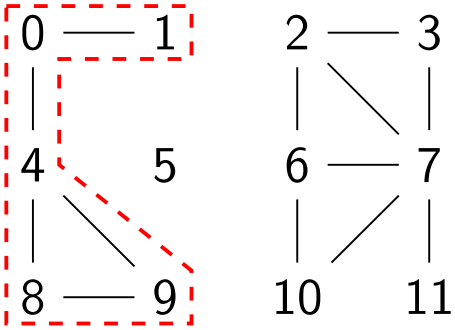
Identifying connected components

- Assign each vertex a component number
- Start BFS/DFS from vertex 0
  - Initialize component number to 0
  - All visited nodes form a connected component

Connected Graph



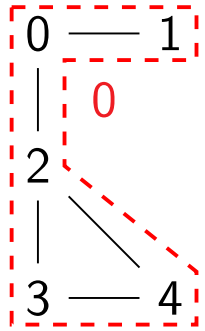
Disconnected Graph



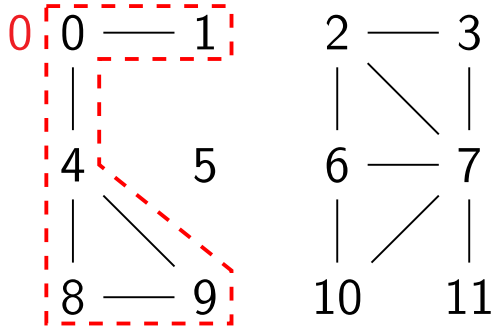
Identifying connected components

- Assign each vertex a component number
- Start BFS/DFS from vertex 0
  - Initialize component number to 0
  - All visited nodes form a connected component
  - Assign each visited node component number 0

Connected Graph



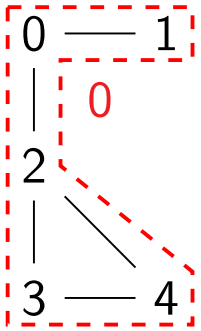
Disconnected Graph



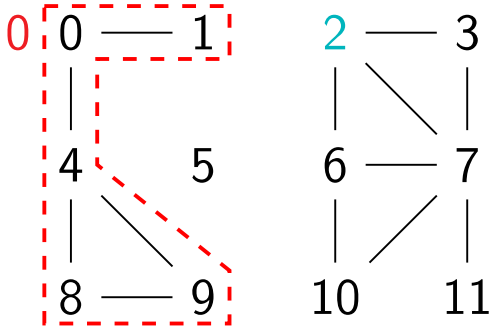
Identifying connected components

- Assign each vertex a component number
- Start BFS/DFS from vertex 0
  - Initialize component number to 0
  - All visited nodes form a connected component
  - Assign each visited node component number 0
- Pick smallest unvisited node j

Connected Graph



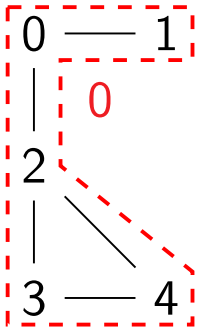
Disconnected Graph



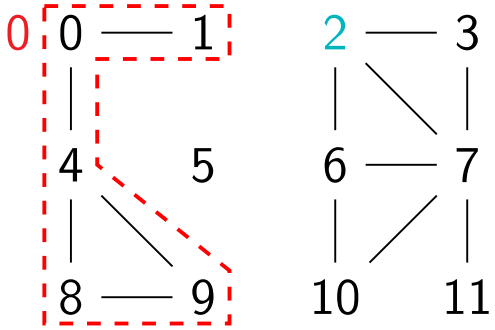
Identifying connected components

- Assign each vertex a component number
- Start BFS/DFS from vertex 0
  - Initialize component number to 0
  - All visited nodes form a connected component
  - Assign each visited node component number 0
- Pick smallest unvisited node j
  - Increment component number to 1

Connected Graph



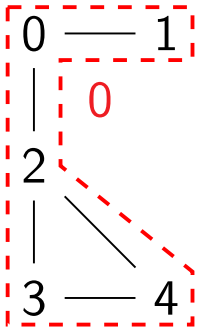
Disconnected Graph



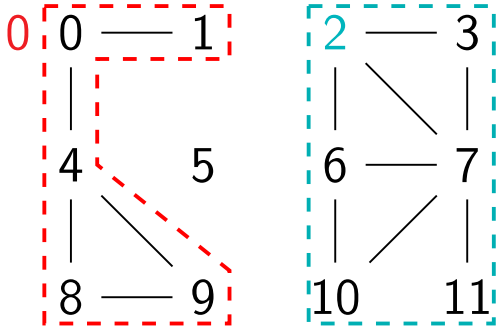
Identifying connected components

- Assign each vertex a component number
- Start BFS/DFS from vertex 0
  - Initialize component number to 0
  - All visited nodes form a connected component
  - Assign each visited node component number 0
- Pick smallest unvisited node j
  - Increment component number to 1
  - Run BFS/DFS from node j

Connected Graph



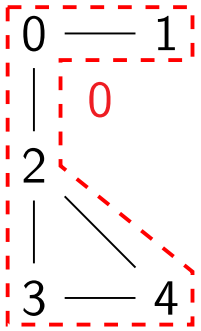
Disconnected Graph



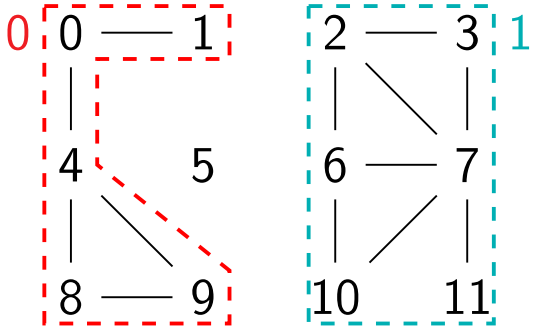
Identifying connected components

- Assign each vertex a component number
- Start BFS/DFS from vertex 0
  - Initialize component number to 0
  - All visited nodes form a connected component
  - Assign each visited node component number 0
- Pick smallest unvisited node j
  - Increment component number to 1
  - Run BFS/DFS from node j
  - Assign each visited node component number 1

Connected Graph



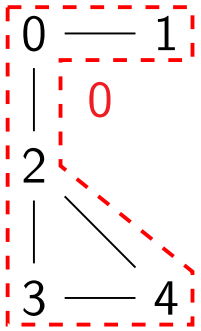
Disconnected Graph



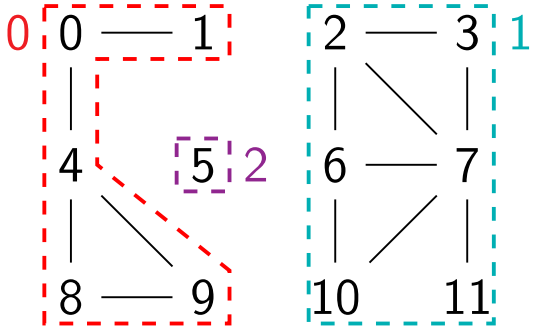
Identifying connected components

- Assign each vertex a component number
- Start BFS/DFS from vertex 0
  - Initialize component number to 0
  - All visited nodes form a connected component
  - Assign each visited node component number 0
- Pick smallest unvisited node j
  - Increment component number to 1
  - Run BFS/DFS from node j
  - Assign each visited node component number 1
- Repeat until all nodes are visited

Connected Graph

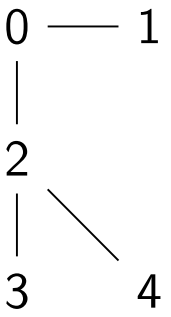


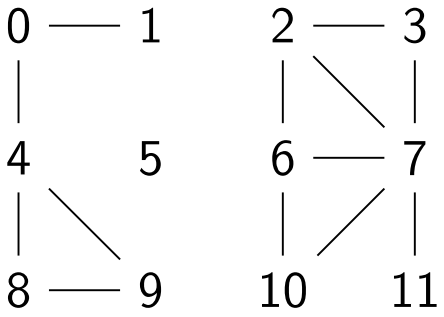
Disconnected Graph



Detecting cycles

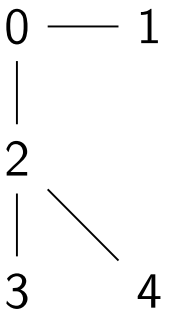
- A cycle is a path (technically, a walk) that starts and ends at the same vertex

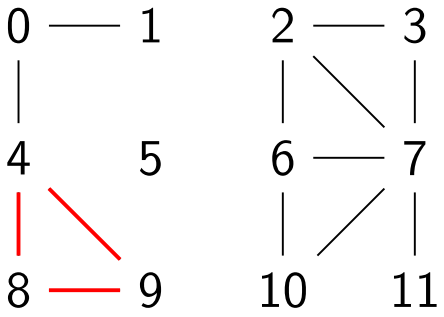




Detecting cycles

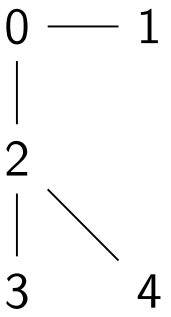
- A cycle is a path (technically, a walk) that starts and ends at the same vertex
  - $4 - 8 - 9 - 4$ is a cycle

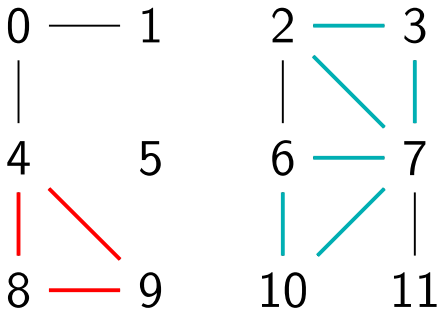




Detecting cycles

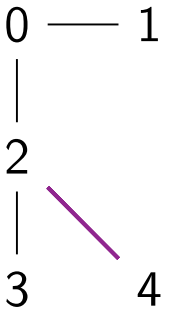
- A cycle is a path (technically, a walk) that starts and ends at the same vertex
  - $4 - 8 - 9 - 4$ is a cycle
  - Cycle may repeat a vertex: $2 - 3 - 7 - 10 - 6 - 7 - 2$

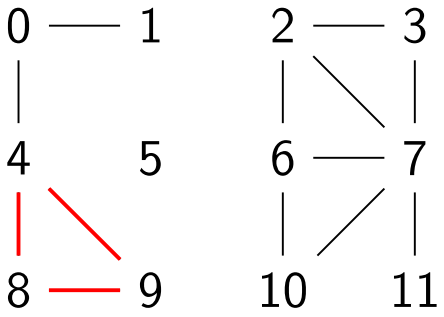




Detecting cycles

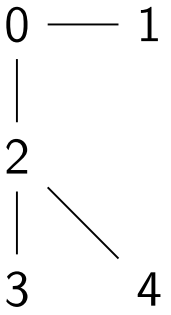
- A cycle is a path (technically, a walk) that starts and ends at the same vertex
  - $4 - 8 - 9 - 4$ is a cycle
  - Cycle may repeat a vertex: $2 - 3 - 7 - 10 - 6 - 7 - 2$
  - Cycle should not repeat edges: $i - j - i$ is not a cycle, e.g., $2 - 4 - 2$

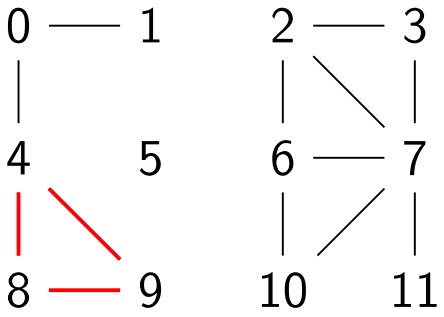




Detecting cycles

- A cycle is a path (technically, a walk) that starts and ends at the same vertex
  - $4 - 8 - 9 - 4$ is a cycle
  - Cycle may repeat a vertex: $2 - 3 - 7 - 10 - 6 - 7 - 2$
  - Cycle should not repeat edges: $i - j - i$ is not a cycle, e.g., $2 - 4 - 2$
  - Simple cycle — only repeated vertices are start and end

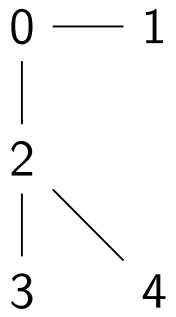




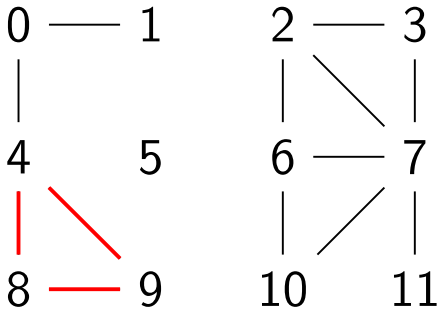
Detecting cycles

- A cycle is a path (technically, a walk) that starts and ends at the same vertex
  - $4 - 8 - 9 - 4$ is a cycle
  - Cycle may repeat a vertex: $2 - 3 - 7 - 10 - 6 - 7 - 2$
  - Cycle should not repeat edges: $i - j - i$ is not a cycle, e.g., $2 - 4 - 2$
  - Simple cycle — only repeated vertices are start and end
- A graph is acyclic if it has no cycles

Acyclic Graph



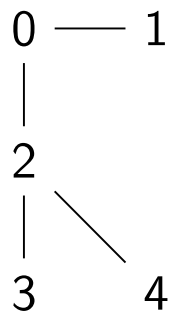
Graph with cycles



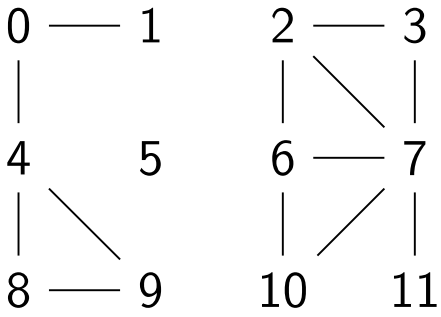
BFS tree

- A tree is a minimally connected graph

Acyclic Graph



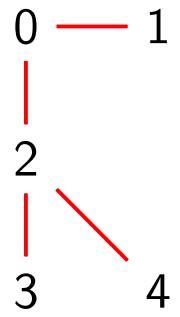
Graph with cycles



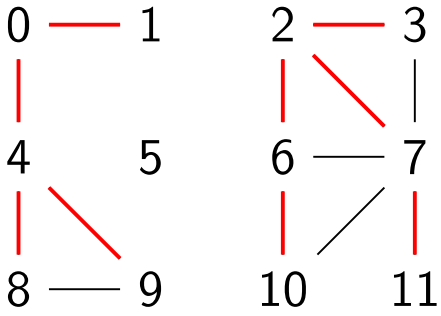
BFS tree

- A tree is a minimally connected graph
- Edges explored by BFS form a tree
  - Technically, one tree per component
  - Collection of trees is a forest

Acyclic Graph



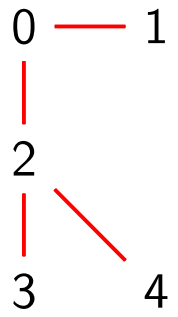
Graph with cycles



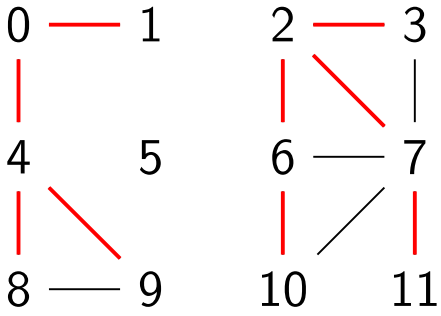
BFS tree

- A tree is a minimally connected graph
- Edges explored by BFS form a tree
  - Technically, one tree per component
  - Collection of trees is a forest
- Facts about trees

Acyclic Graph



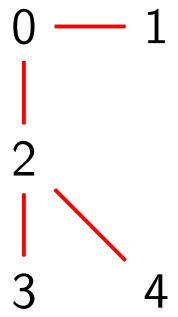
Graph with cycles



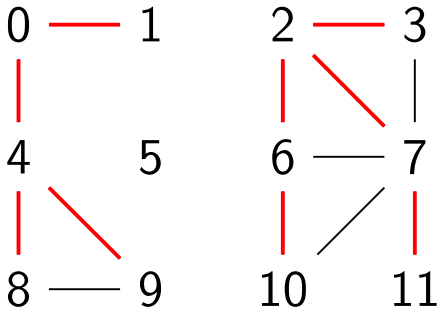
BFS tree

- A tree is a minimally connected graph
- Edges explored by BFS form a tree
  - Technically, one tree per component
  - Collection of trees is a forest
- Facts about trees
  - A tree on n vertices has $n - 1$ edges

Acyclic Graph



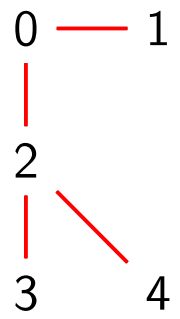
Graph with cycles



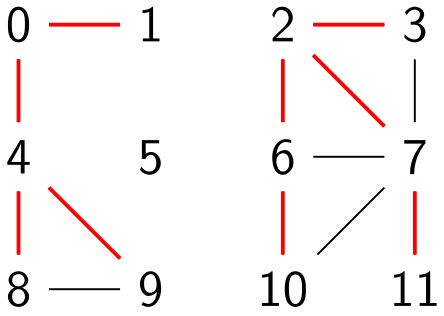
BFS tree

- A tree is a minimally connected graph
- Edges explored by BFS form a tree
  - Technically, one tree per component
  - Collection of trees is a forest
- Facts about trees
  - A tree on n vertices has $n - 1$ edges
  - A tree is acyclic

Acyclic Graph



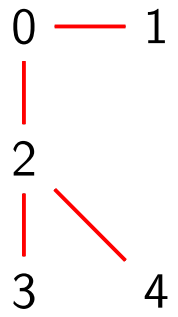
Graph with cycles



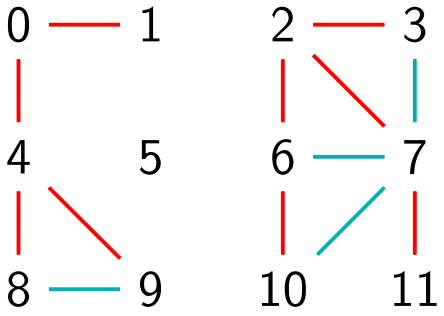
BFS tree

- A tree is a minimally connected graph
- Edges explored by BFS form a tree
  - Technically, one tree per component
  - Collection of trees is a forest
- Facts about trees
  - A tree on n vertices has $n - 1$ edges
  - A tree is acyclic
- Any non-tree edge creates a cycle
  - Detect cycles by searching for non-tree edges

Acyclic Graph

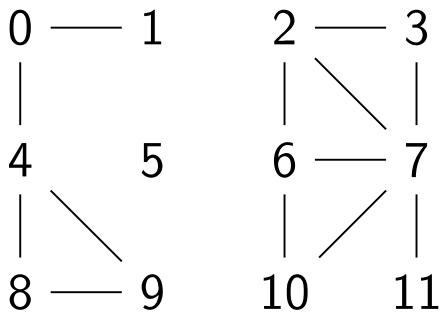


Graph with cycles



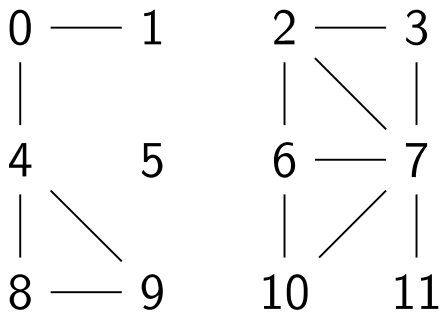
DFS tree

- Maintain a DFS counter, initially 0



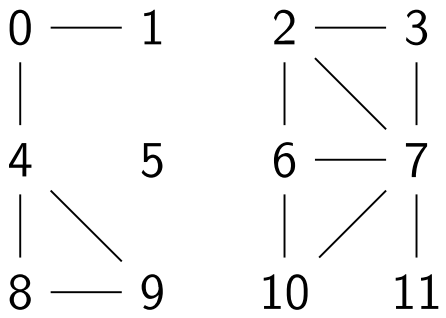
DFS tree

- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node



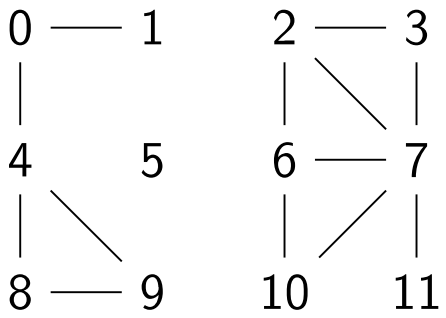
DFS tree

- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)



DFS tree

- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

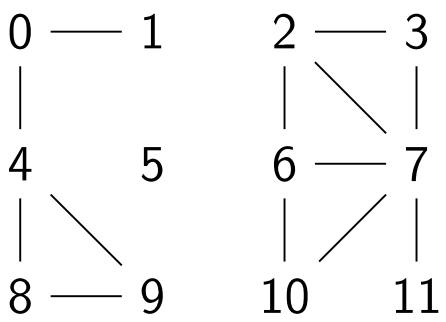


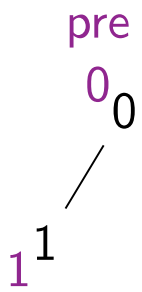
pre

0 0

DFS tree

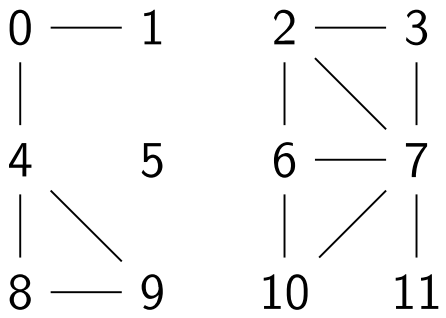
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

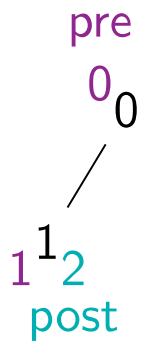




DFS tree

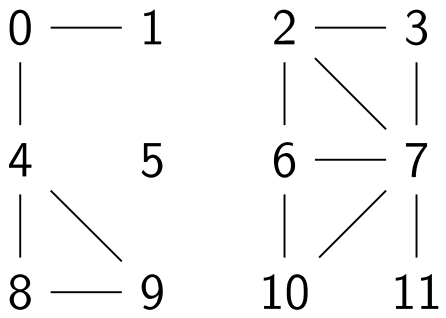
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

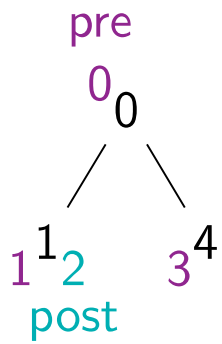




DFS tree

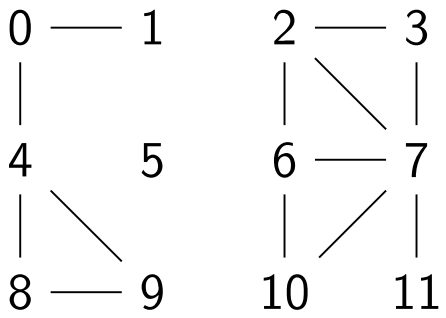
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

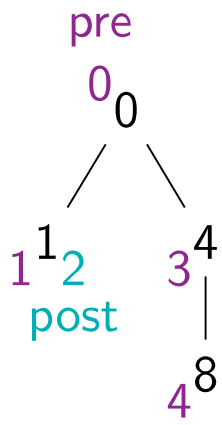




DFS tree

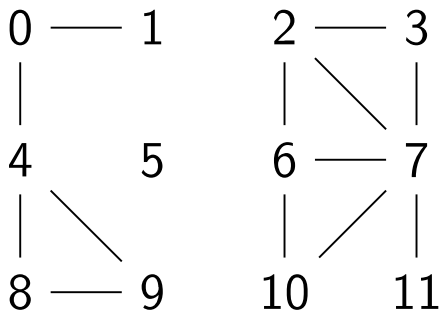
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

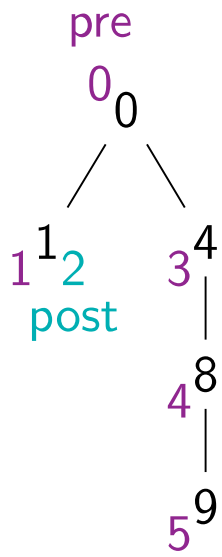




DFS tree

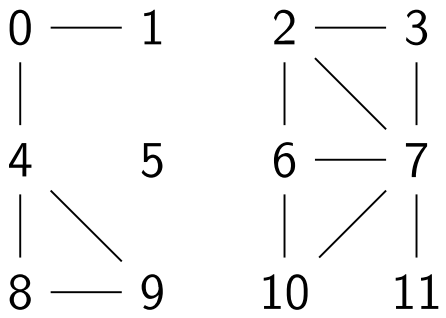
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

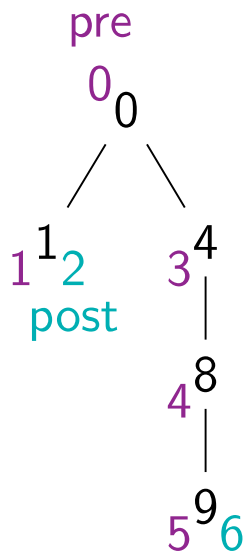




DFS tree

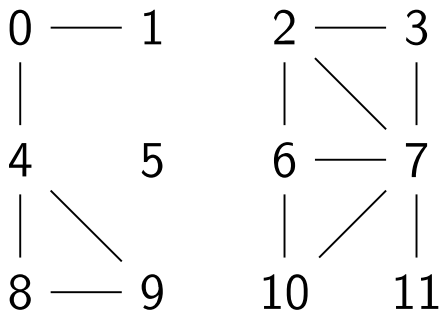
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

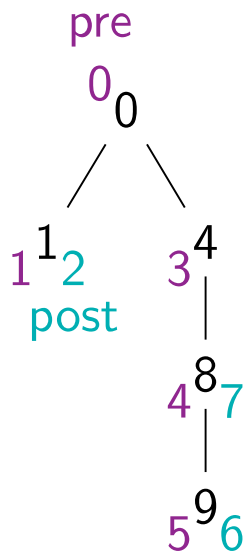




DFS tree

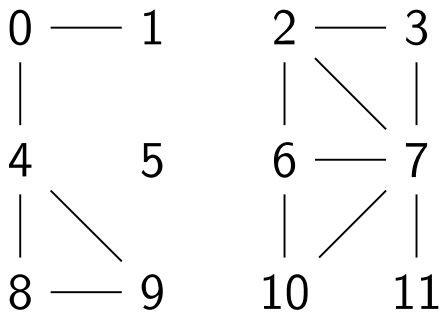
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

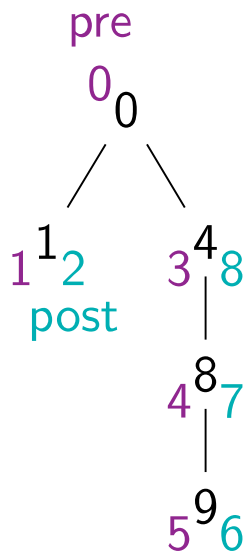




DFS tree

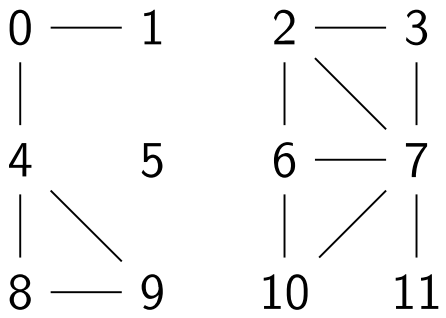
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

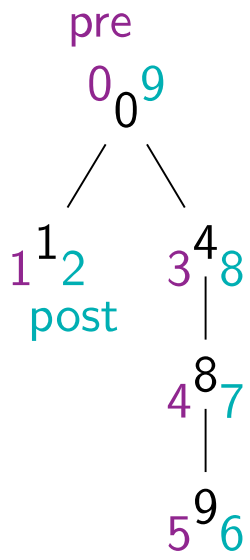




DFS tree

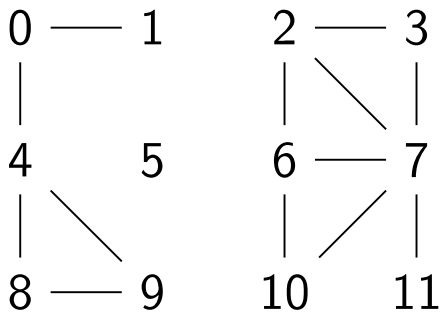
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

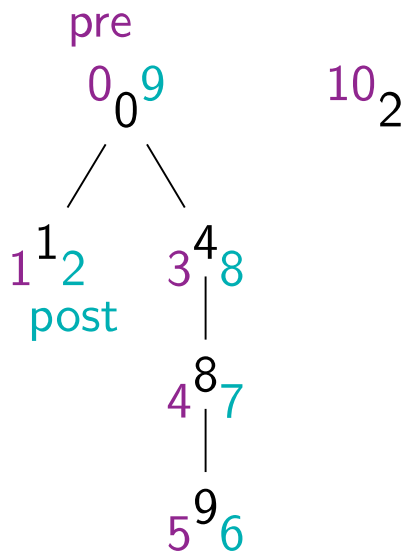




DFS tree

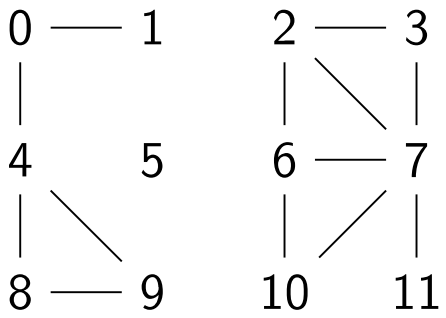
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

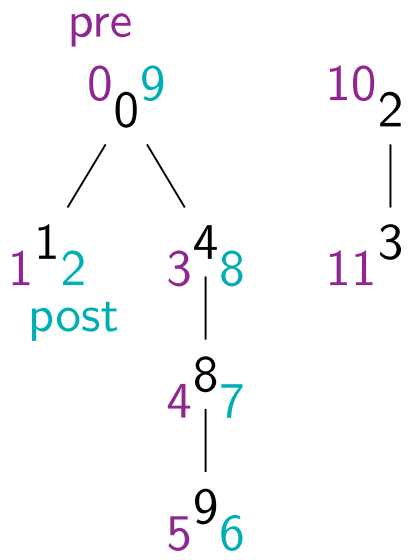




DFS tree

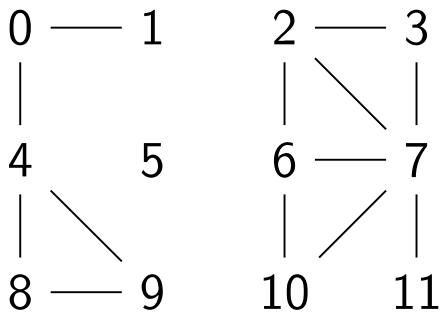
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

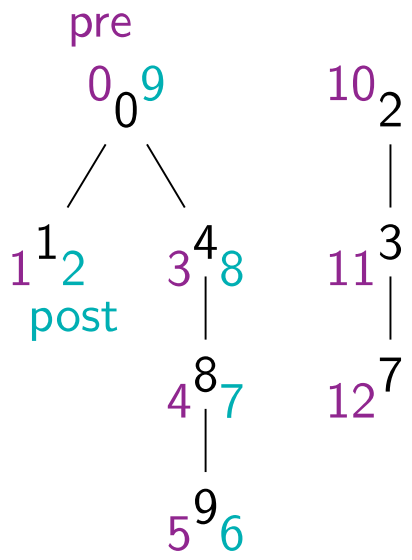




DFS tree

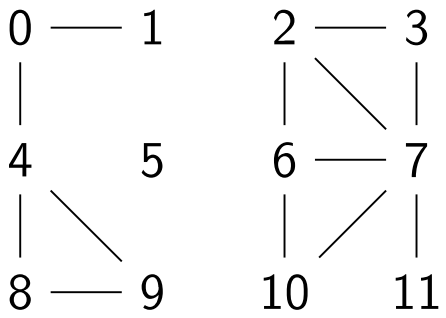
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

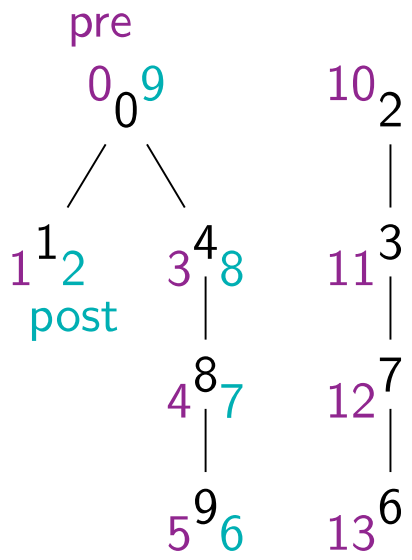




DFS tree

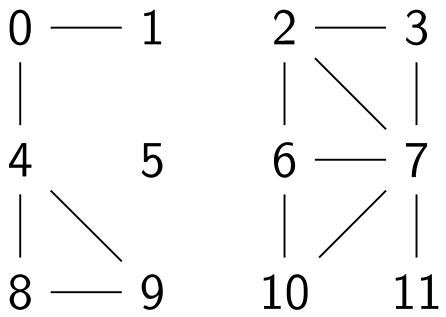
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

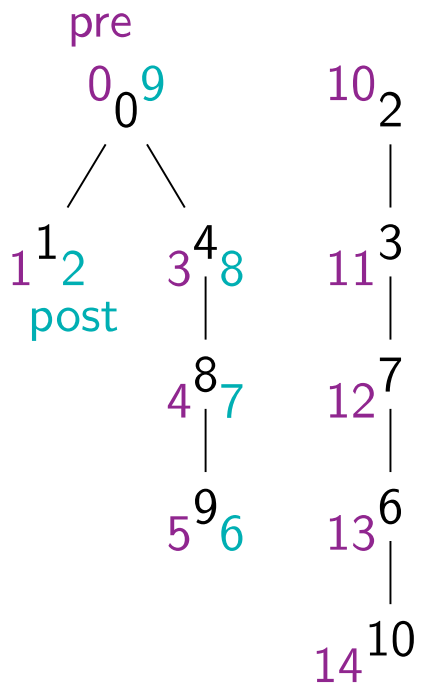




DFS tree

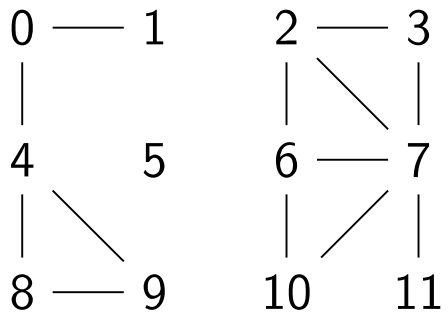
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

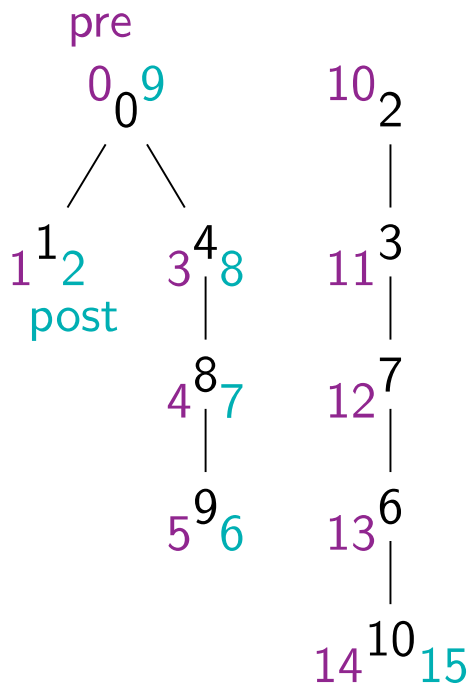




DFS tree

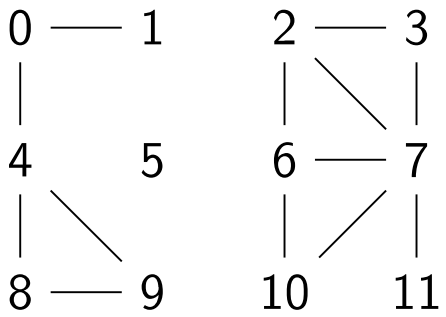
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

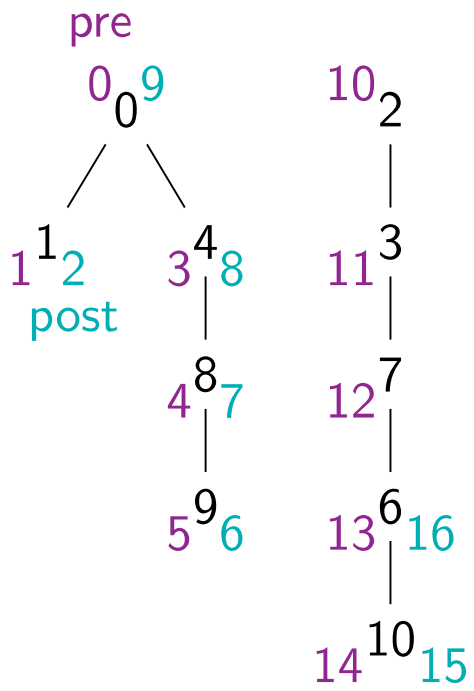




DFS tree

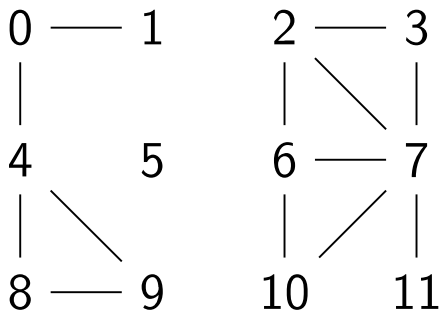
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

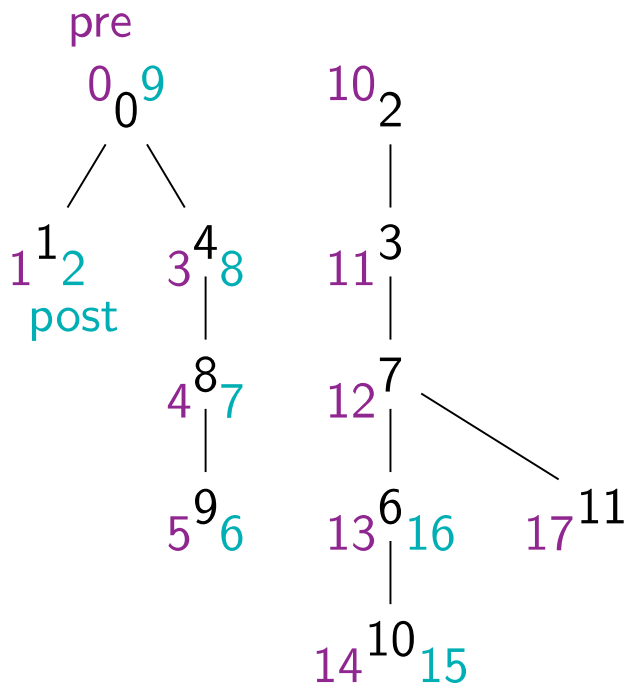




DFS tree

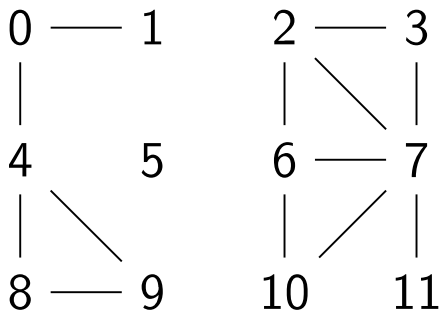
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

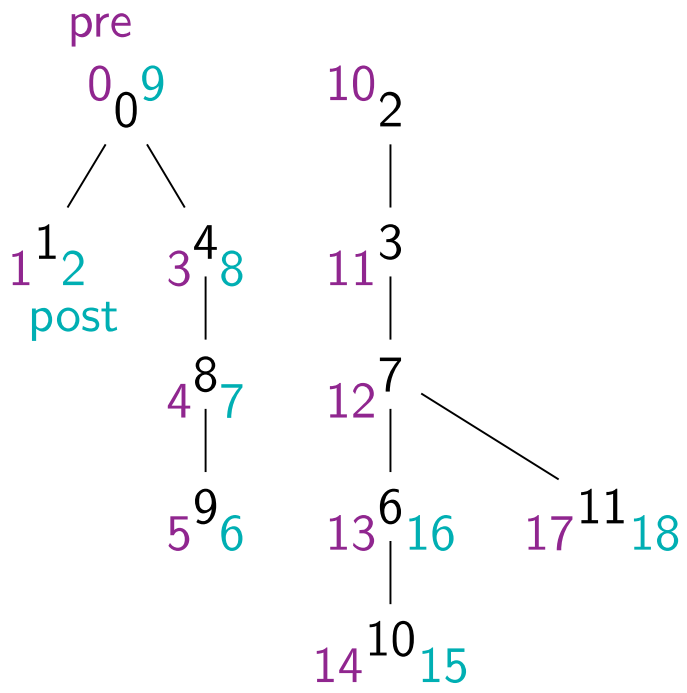




DFS tree

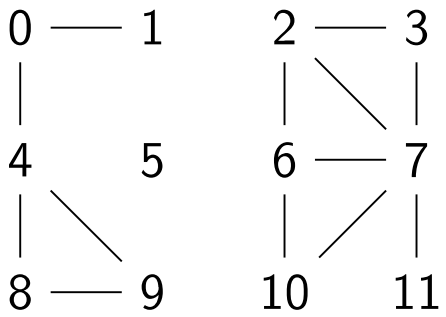
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

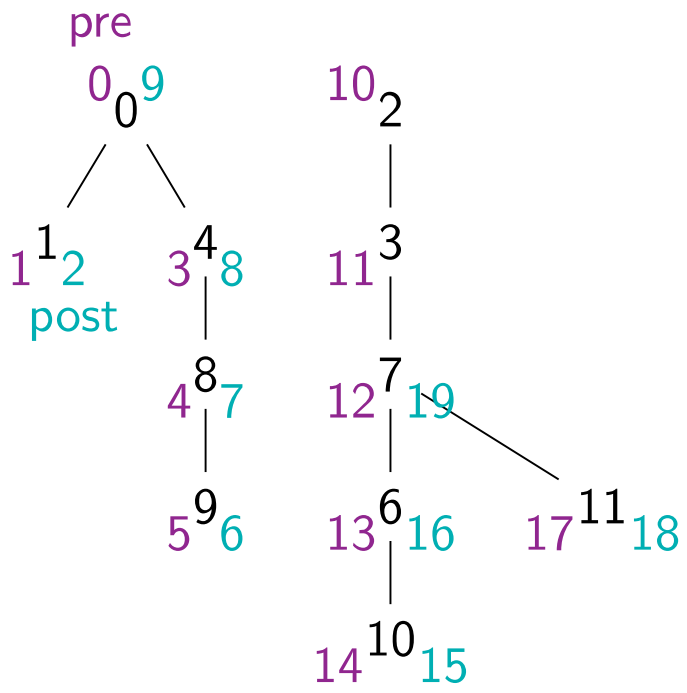




DFS tree

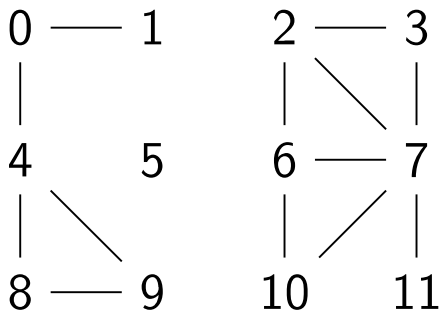
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

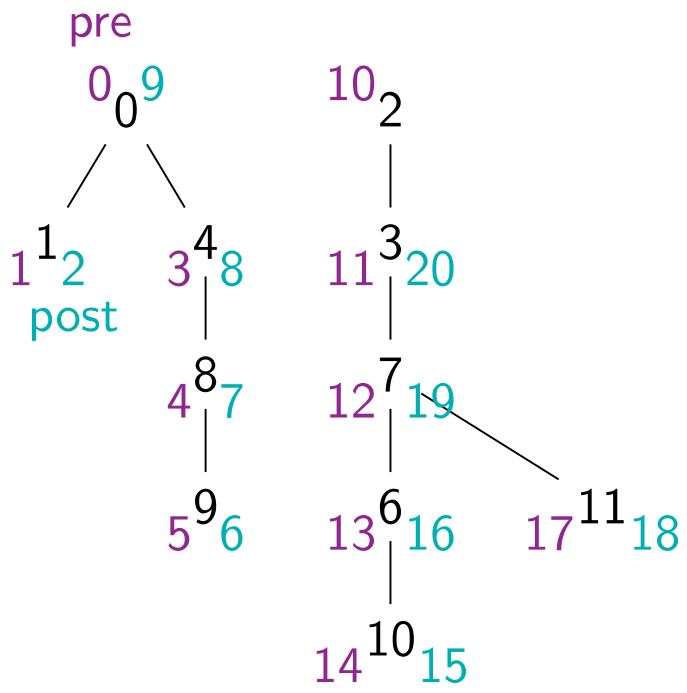




DFS tree

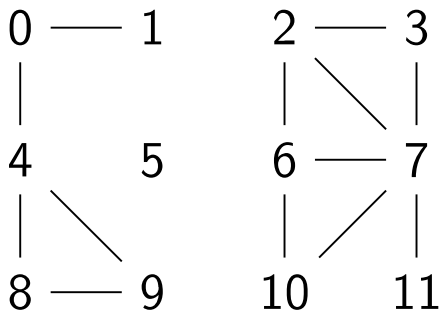
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

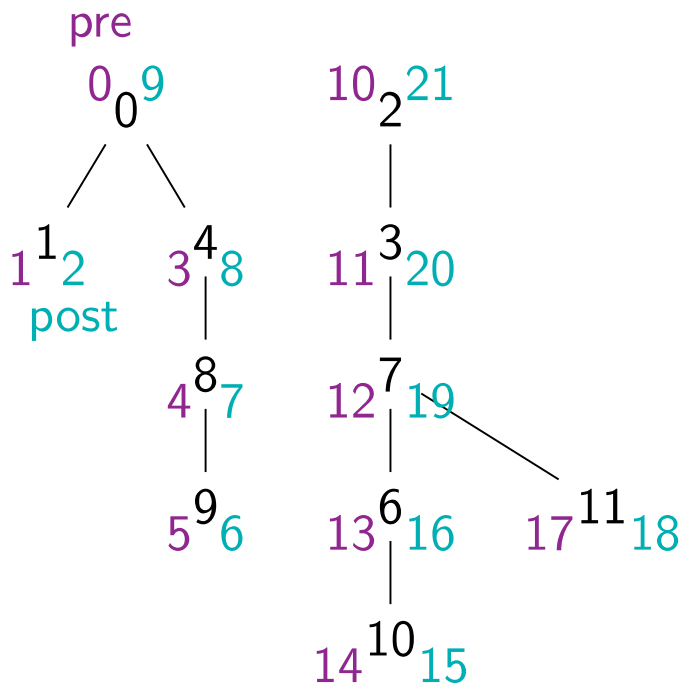




DFS tree

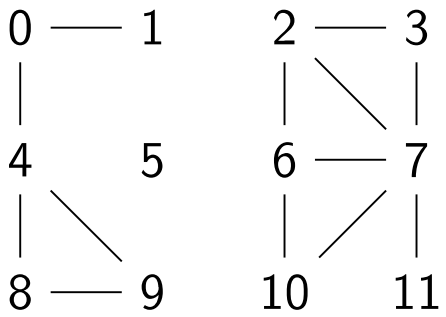
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

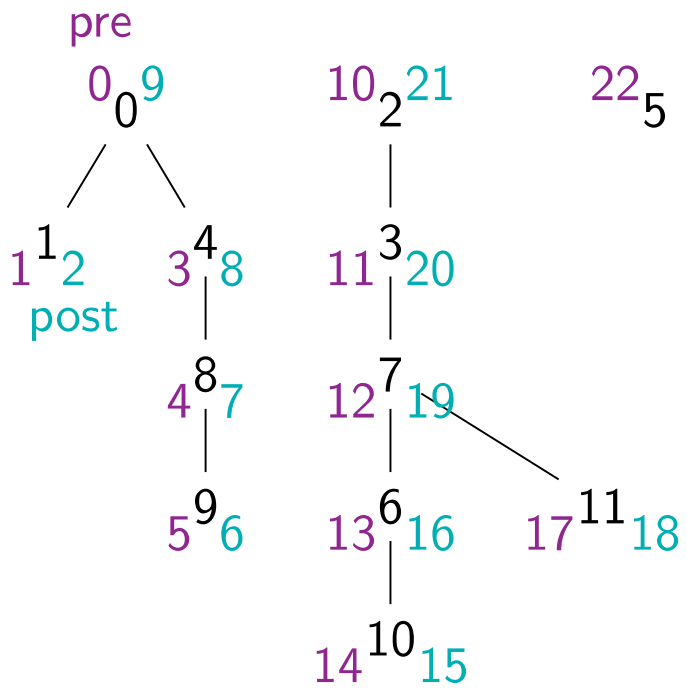




DFS tree

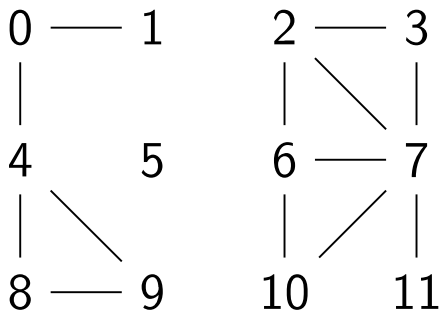
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)

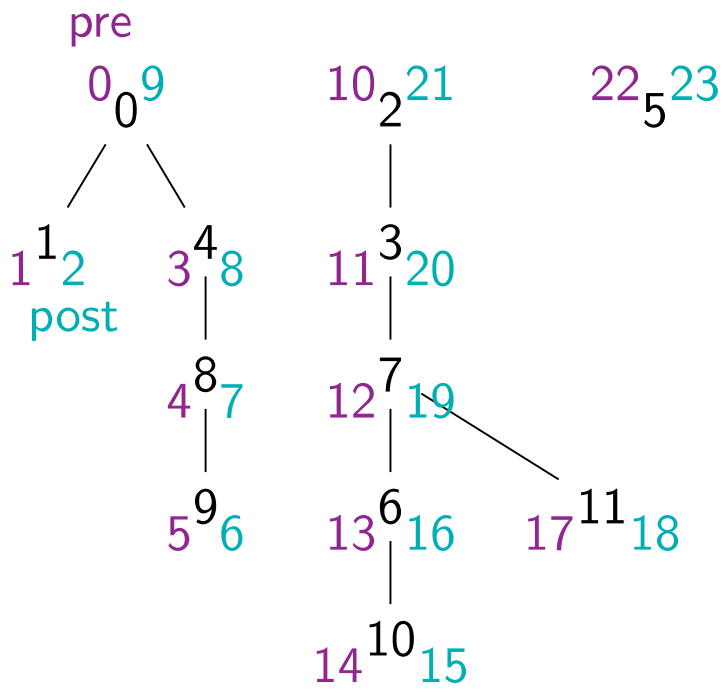




DFS tree

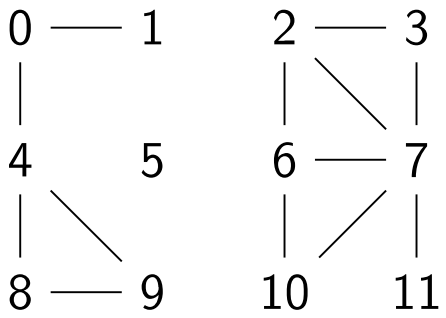
- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)





DFS tree

- Maintain a DFS counter, initially 0
- Increment counter each time we start and finish exploring a node
- Each vertex is assigned an entry number (pre) and exit number (post)



- As before, non-tree edges generate cycles