# Graph Traversals II

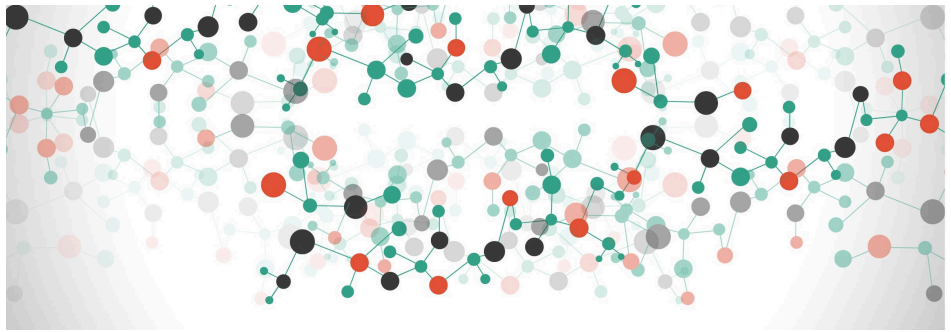
Instructor: Meng-Fen Chiang

COMPCSI220: WEEK 9

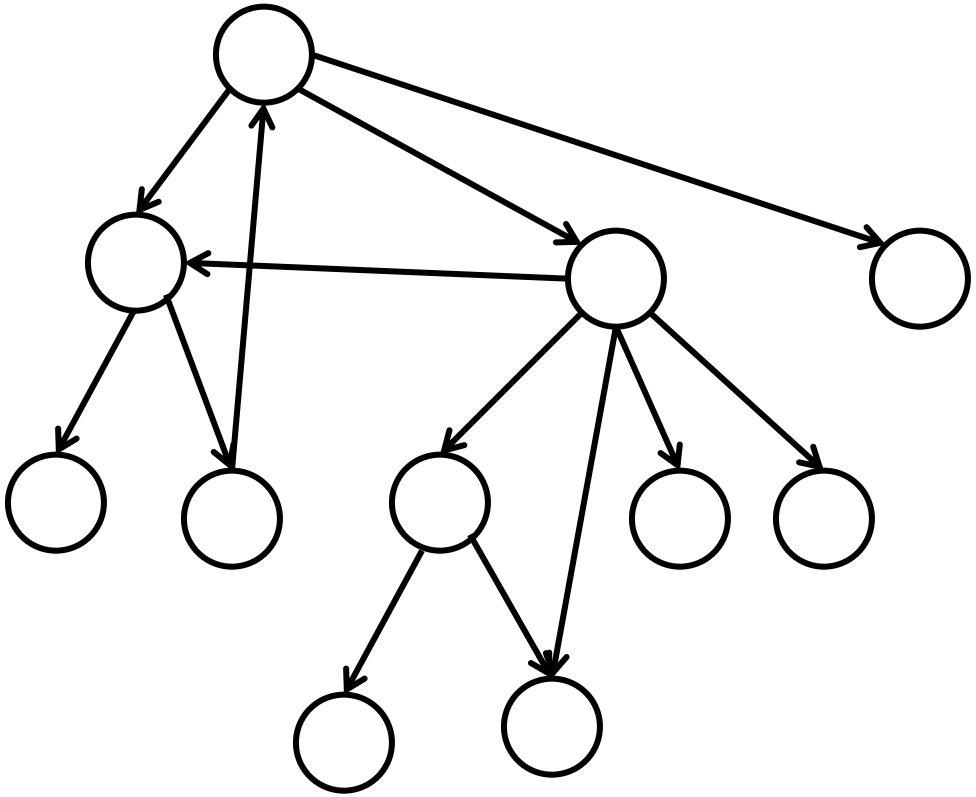

Slides adapted from Mark Wilson, Georgy Gimel'farb, Simone Linz and Tanya Gvozdeva

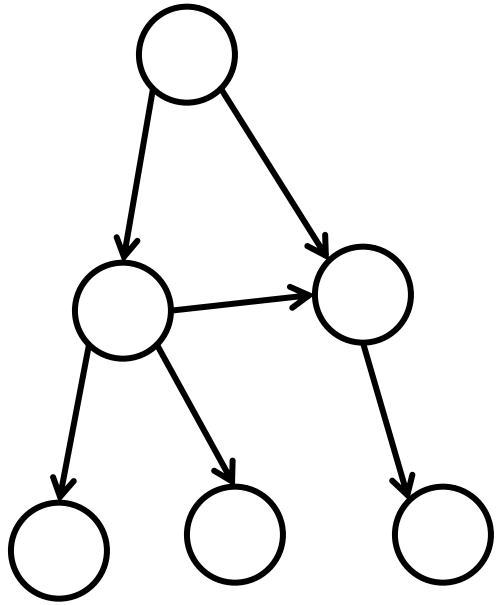
# OUTLINE

- Graph Traversal Algorithms
  - Depth-first Search (DFS)
  - Breadth-first Search (BFS)
  - Priority-first Search (PFS)
- Implementation
  - Stack – DFS
  - Queue – BFS
  - Priority Queues – PFS

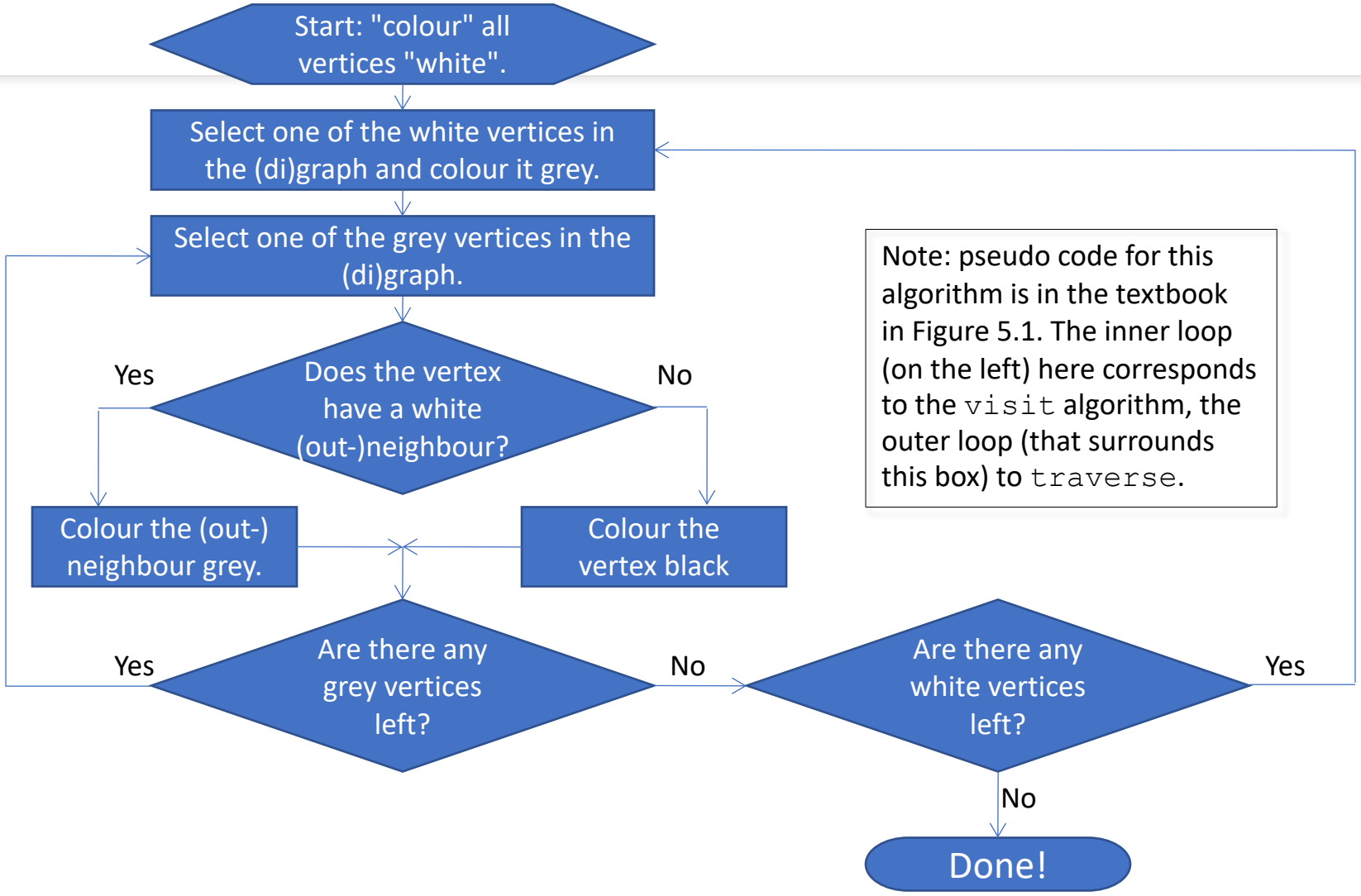


# Graph Traversals





# Graph Traversal Algorithm



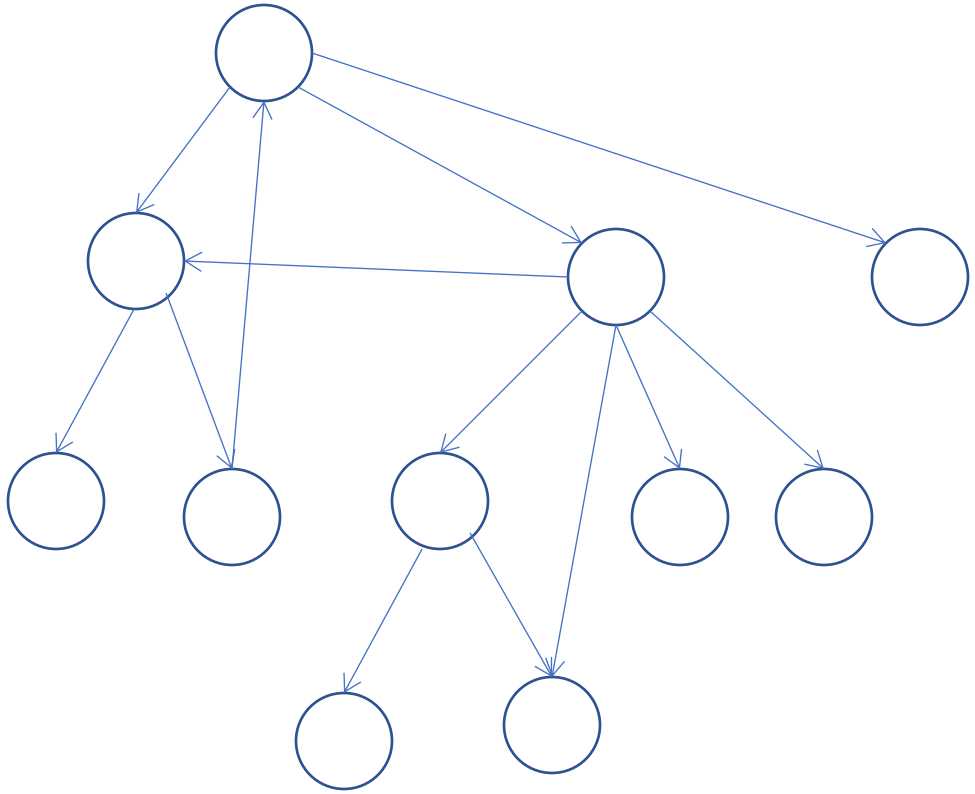

Note: pseudo code for this algorithm is in the textbook in Figure 5.1. The inner loop (on the left) here corresponds to the `visit` algorithm, the outer loop (that surrounds this box) to `traverse`.

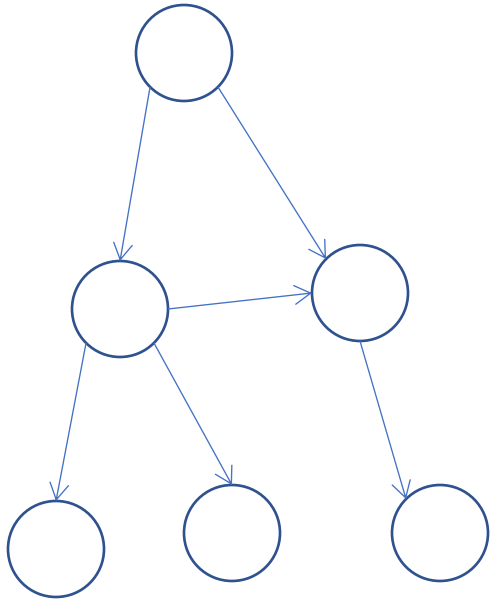
# Depth-first Search Algorithm (DFS)

- DFS is a specific implementation of our fundamental graph traversal algorithm (also known as depth-first traversal)

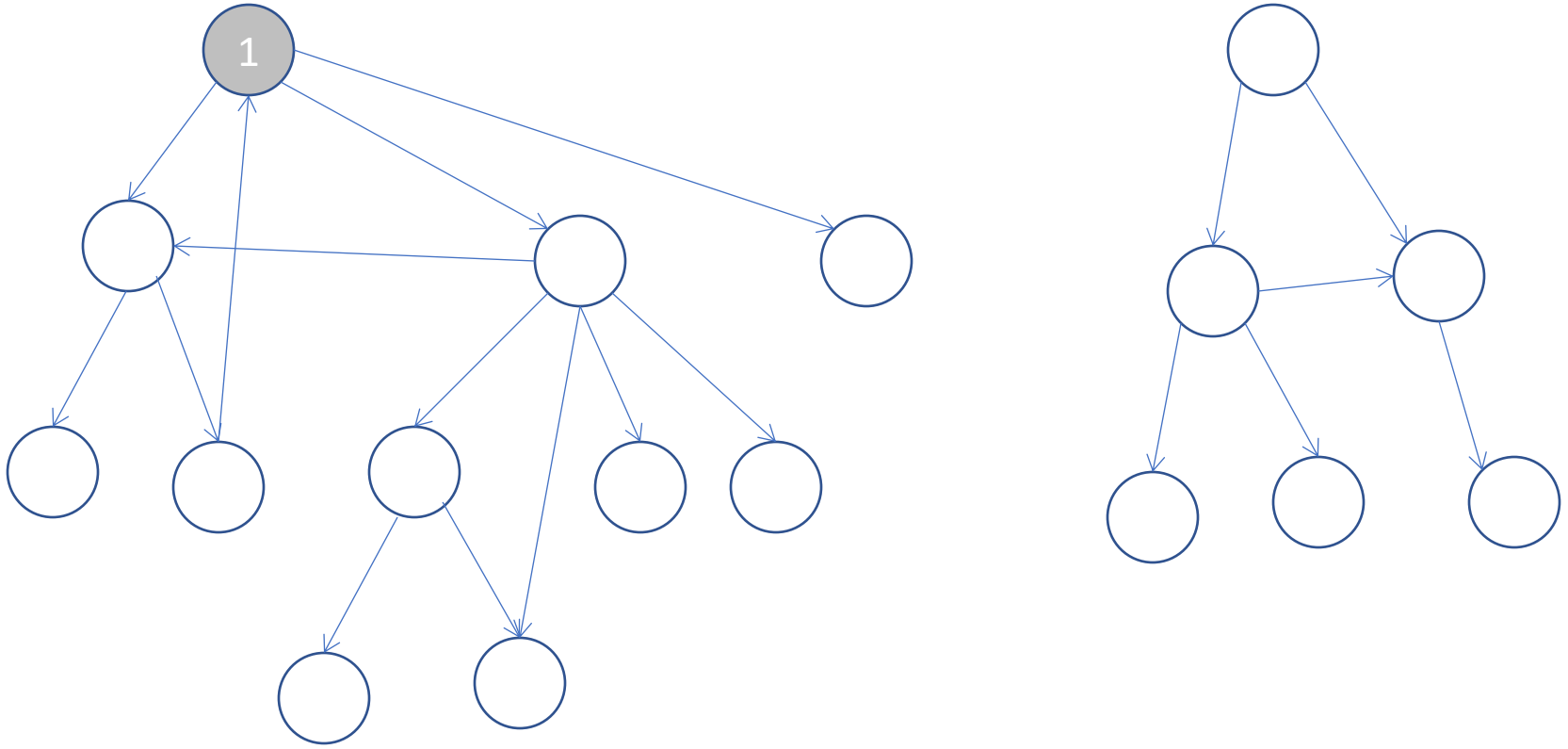
- It specifies that we select the next grey vertex to pick as the youngest remaining grey vertex.

# DFS Traversal

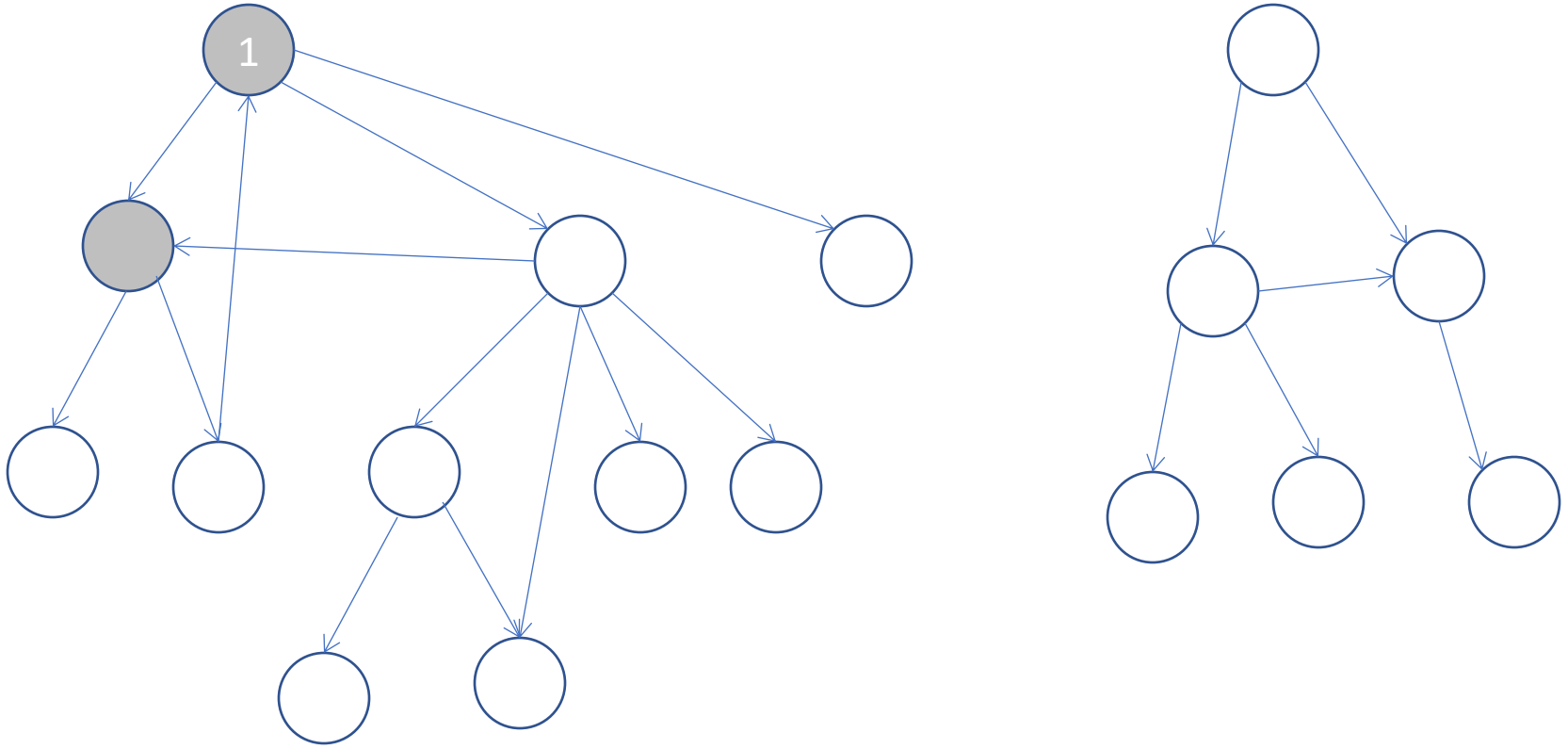




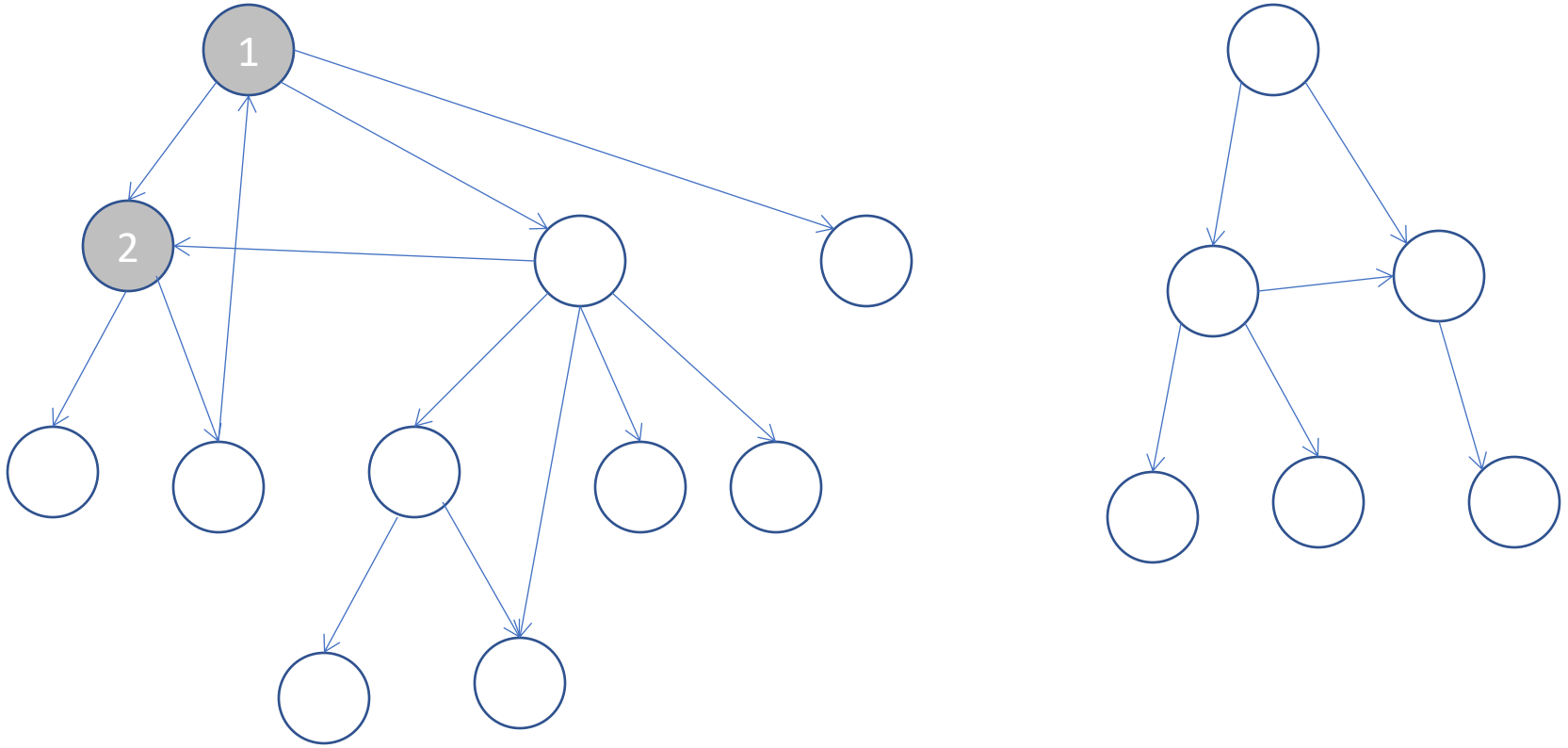
# DFS Traversal



# DFS Traversal

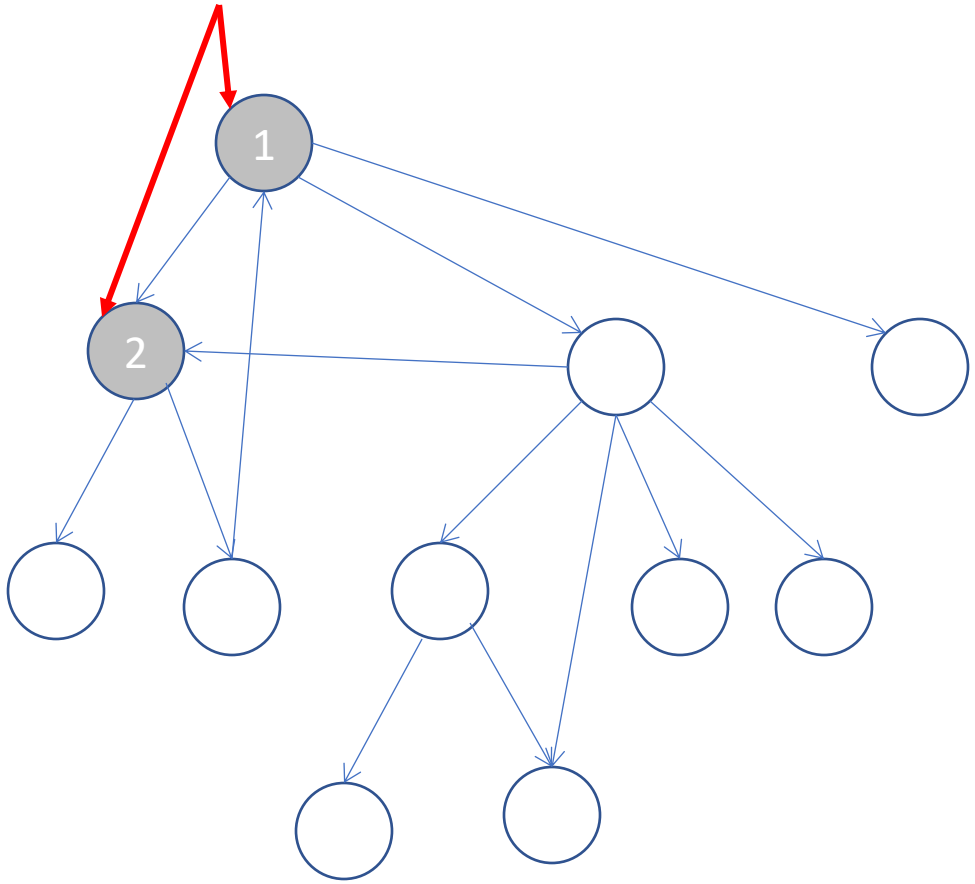


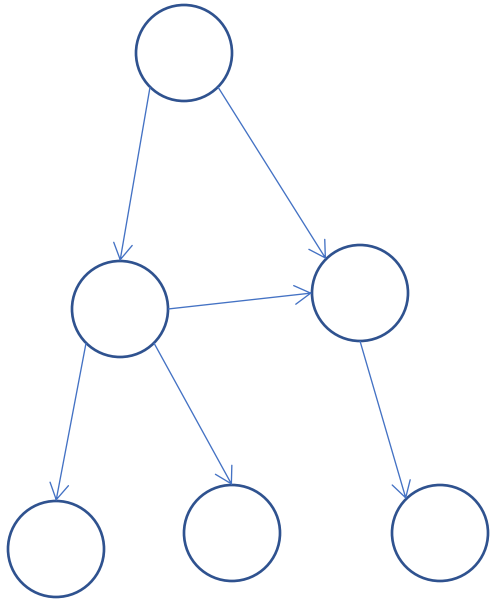
# DFS Traversal



# DFS Traversal

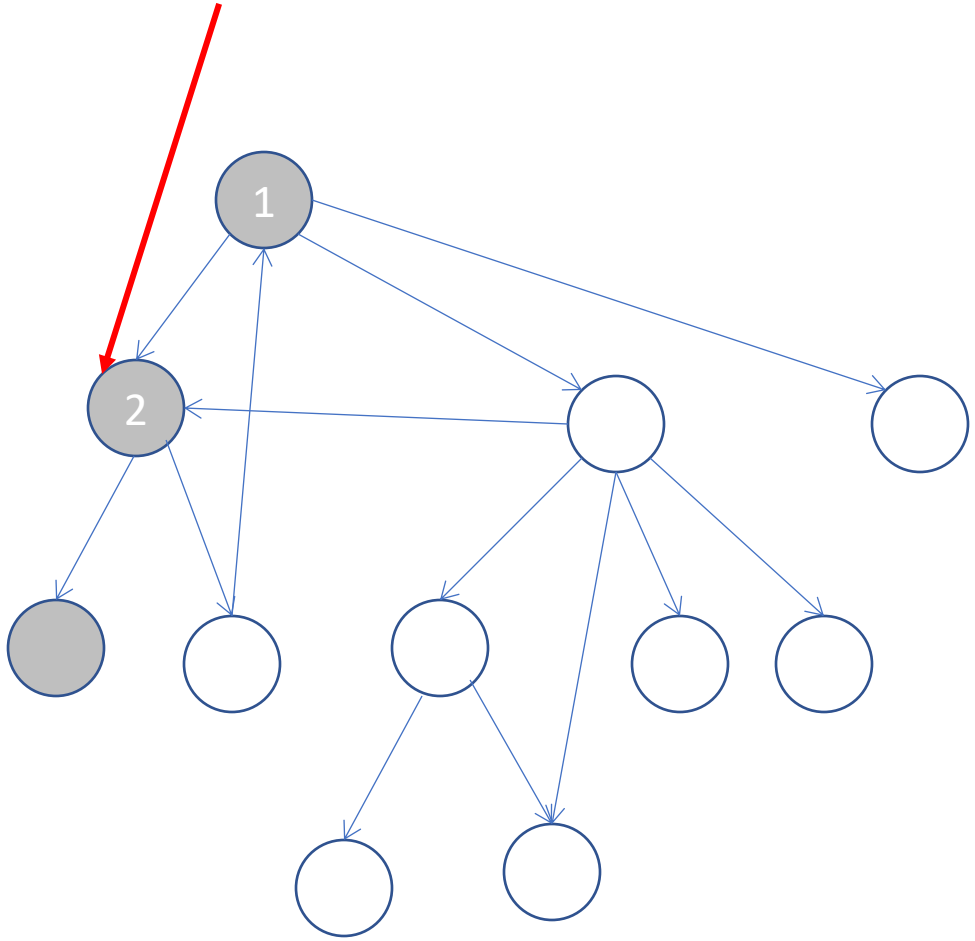
Two grey to chose from…which one?

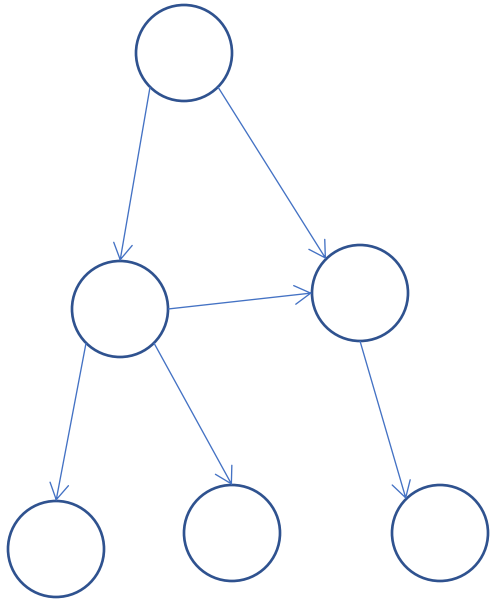


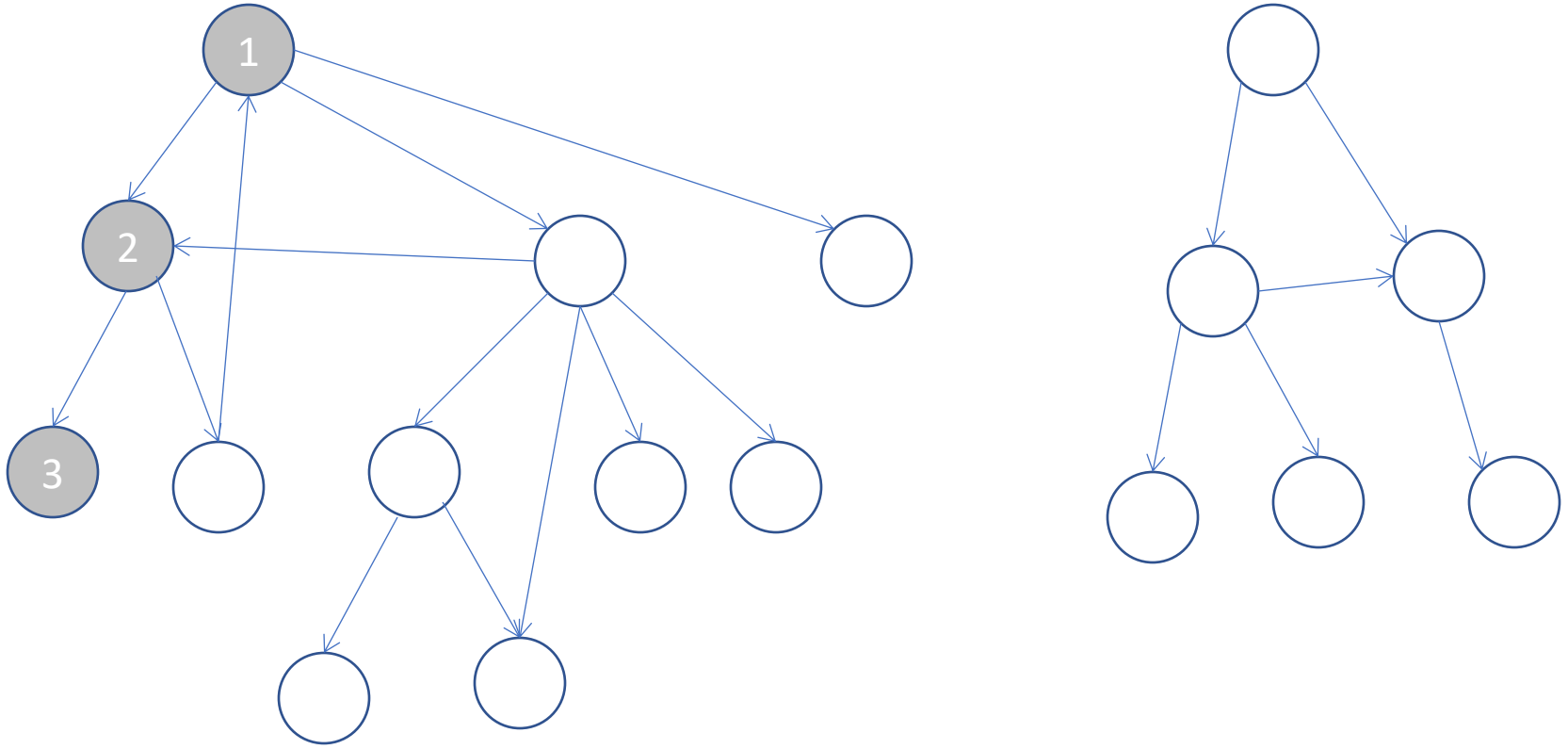
# DFS Traversal

2 is the “youngest”



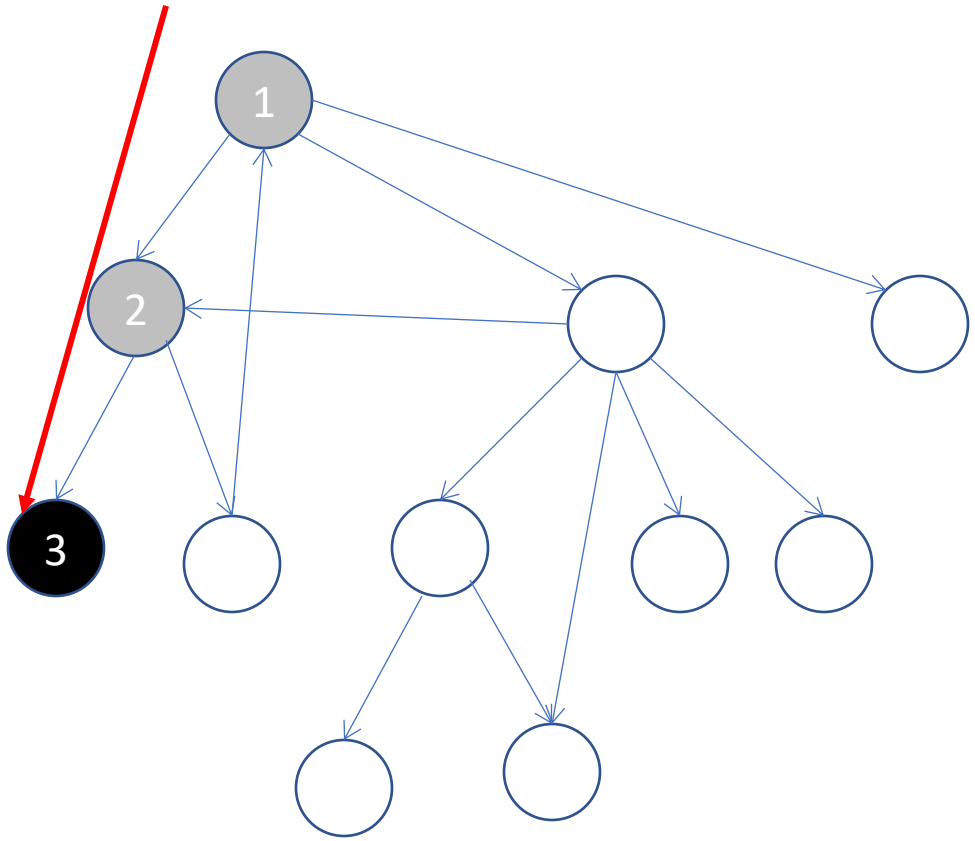



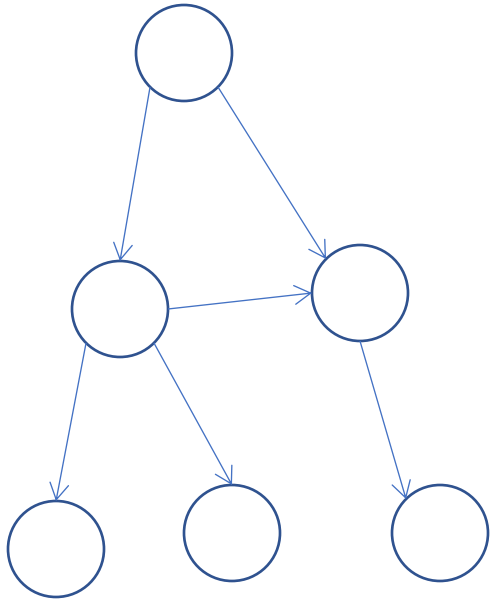
# DFS Traversal



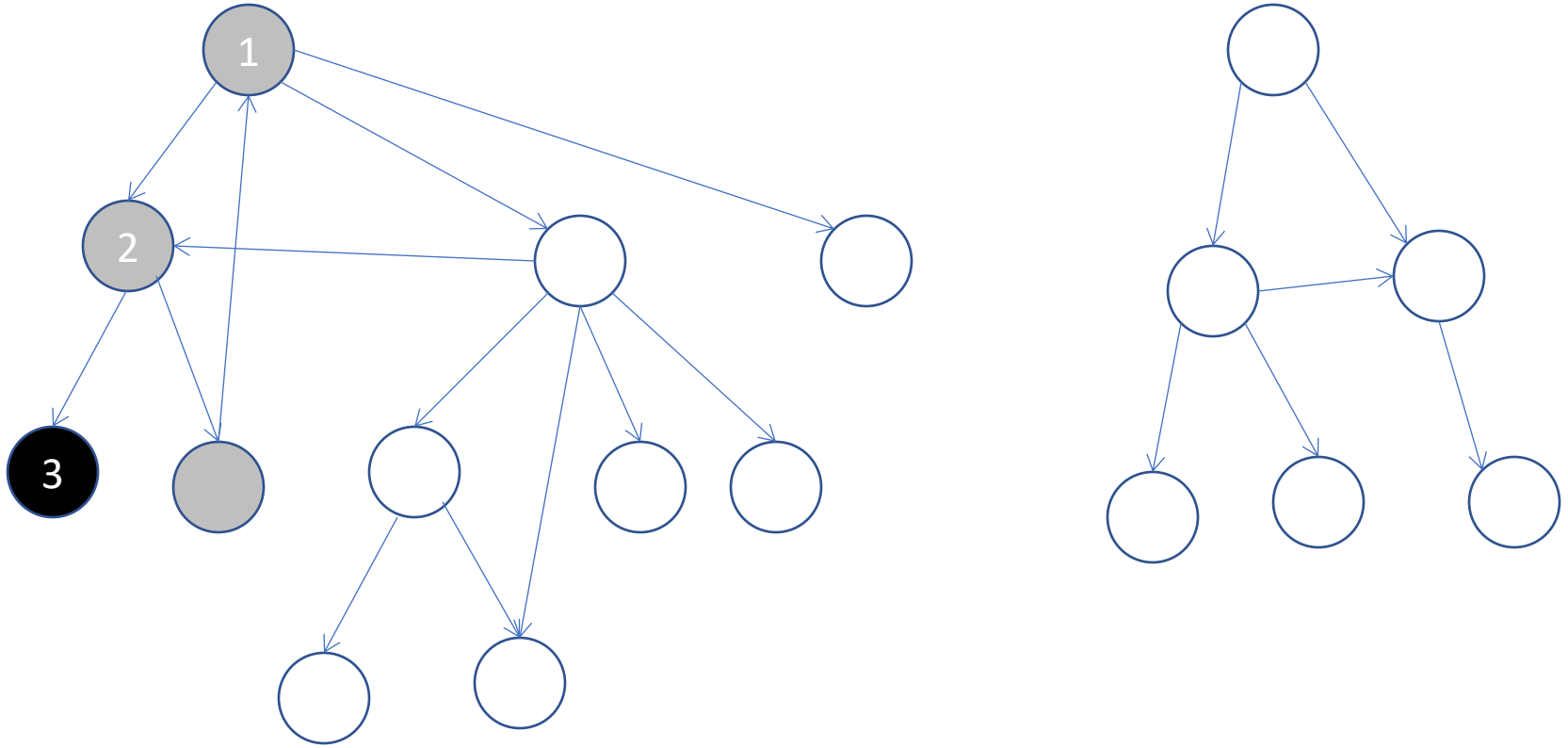
# DFS Traversal

Node 3 does not have anything to offer so it turns black

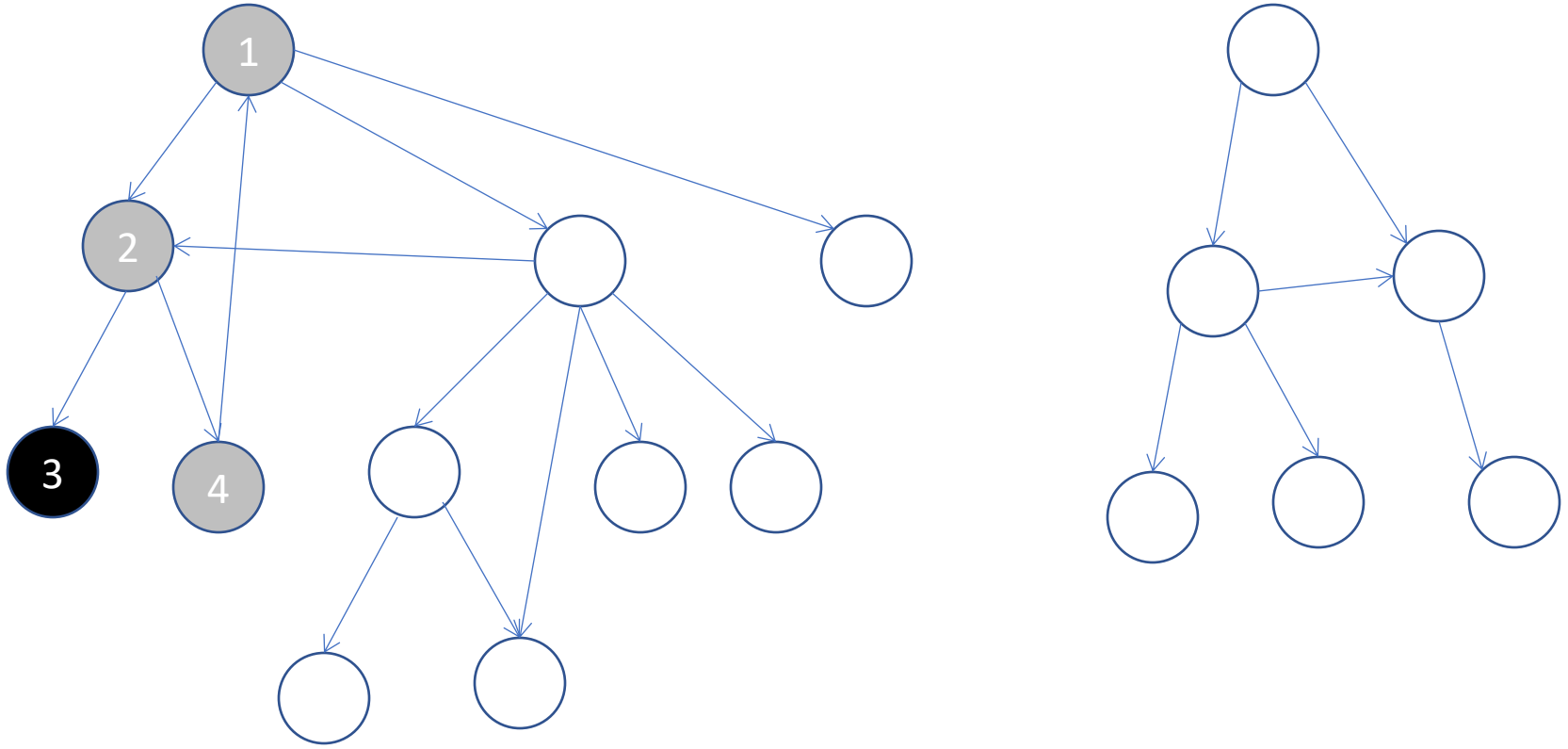


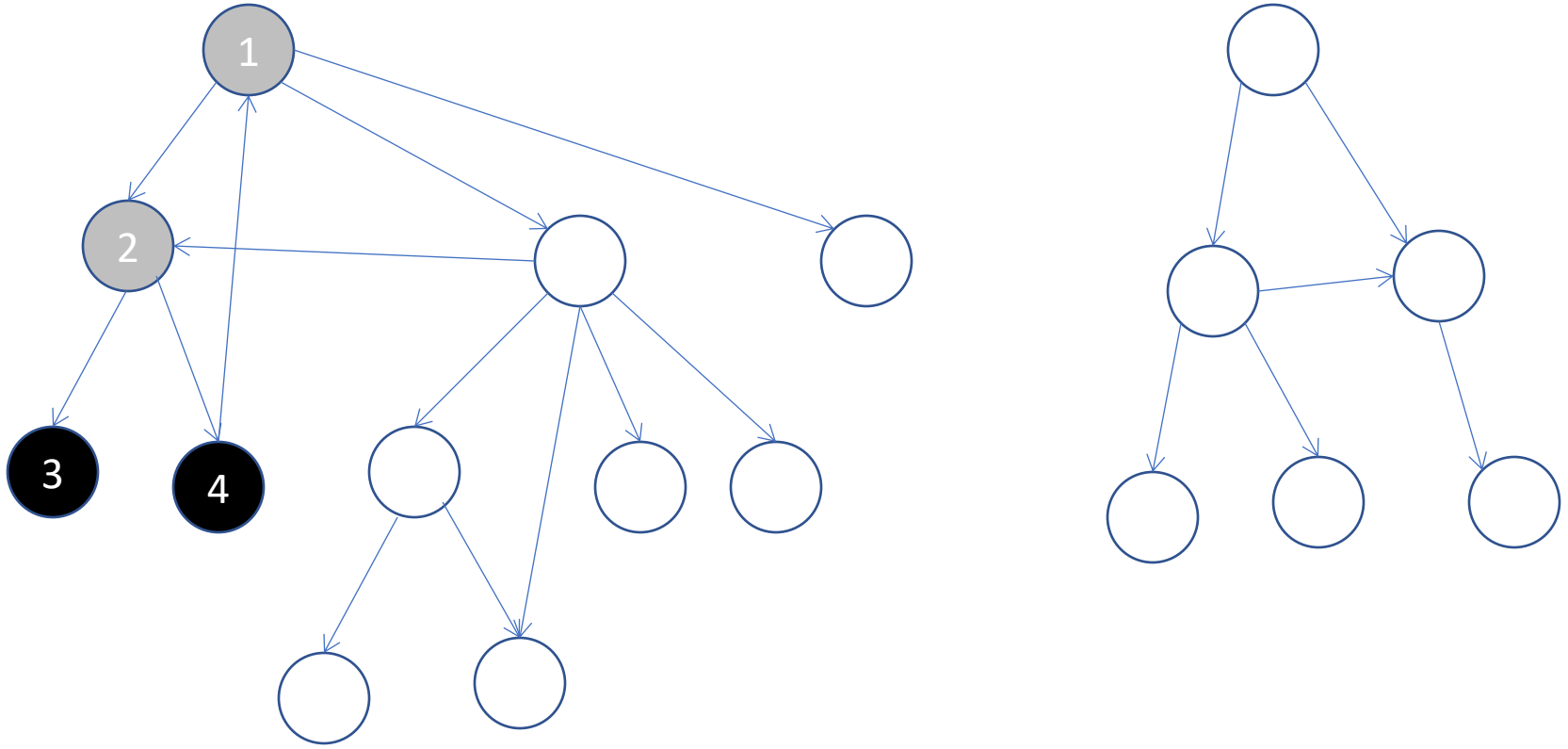
# DFS Traversal



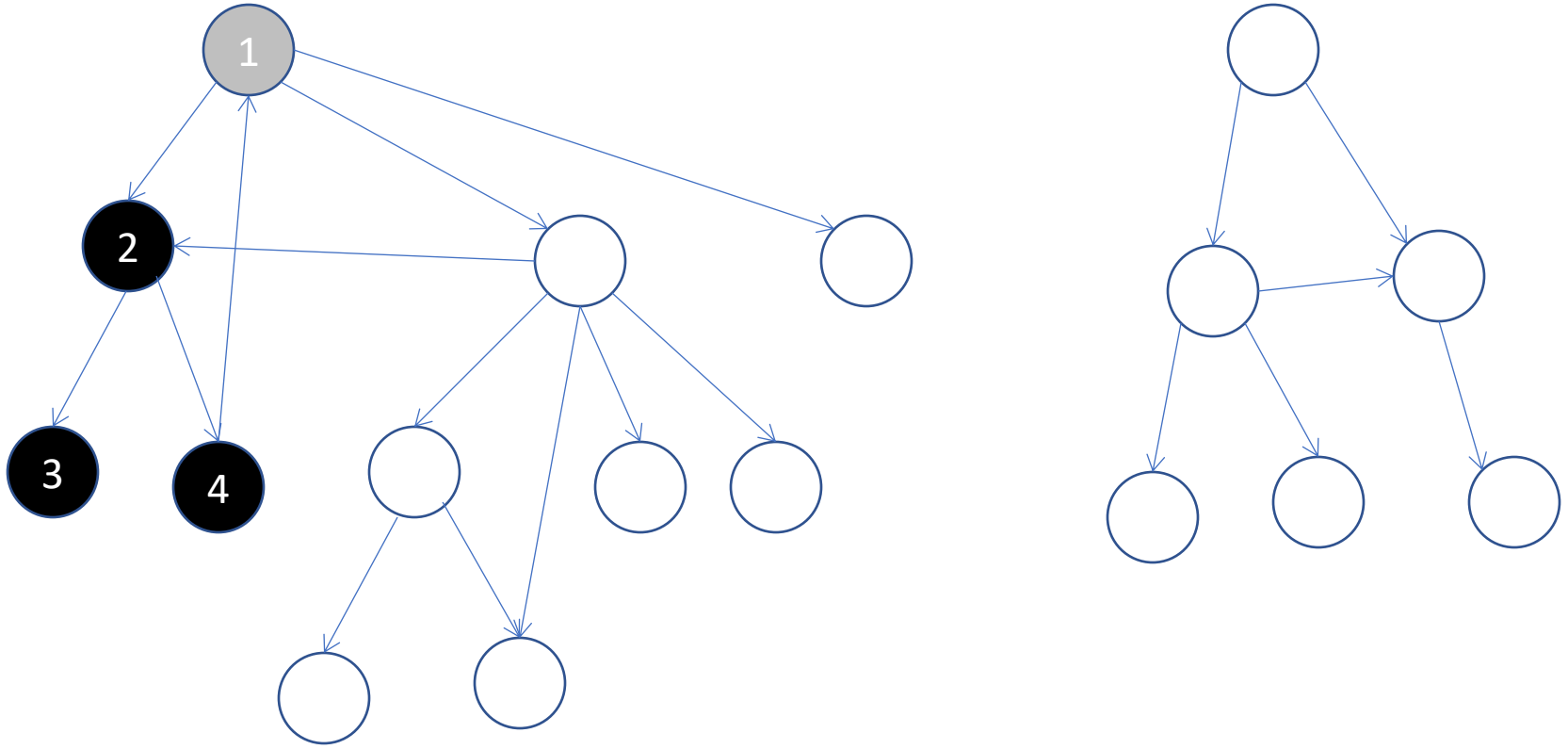
# DFS Traversal



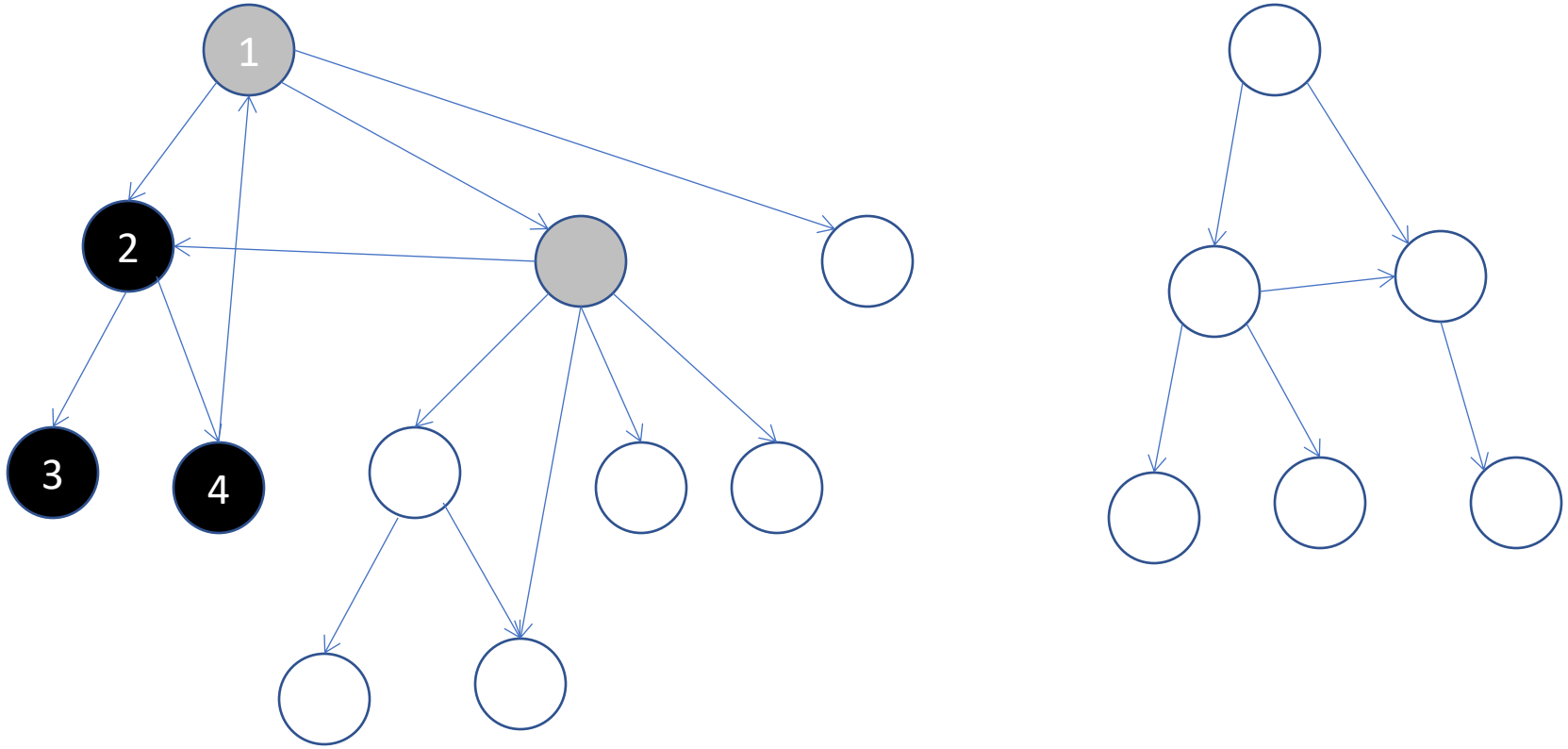
# DFS Traversal



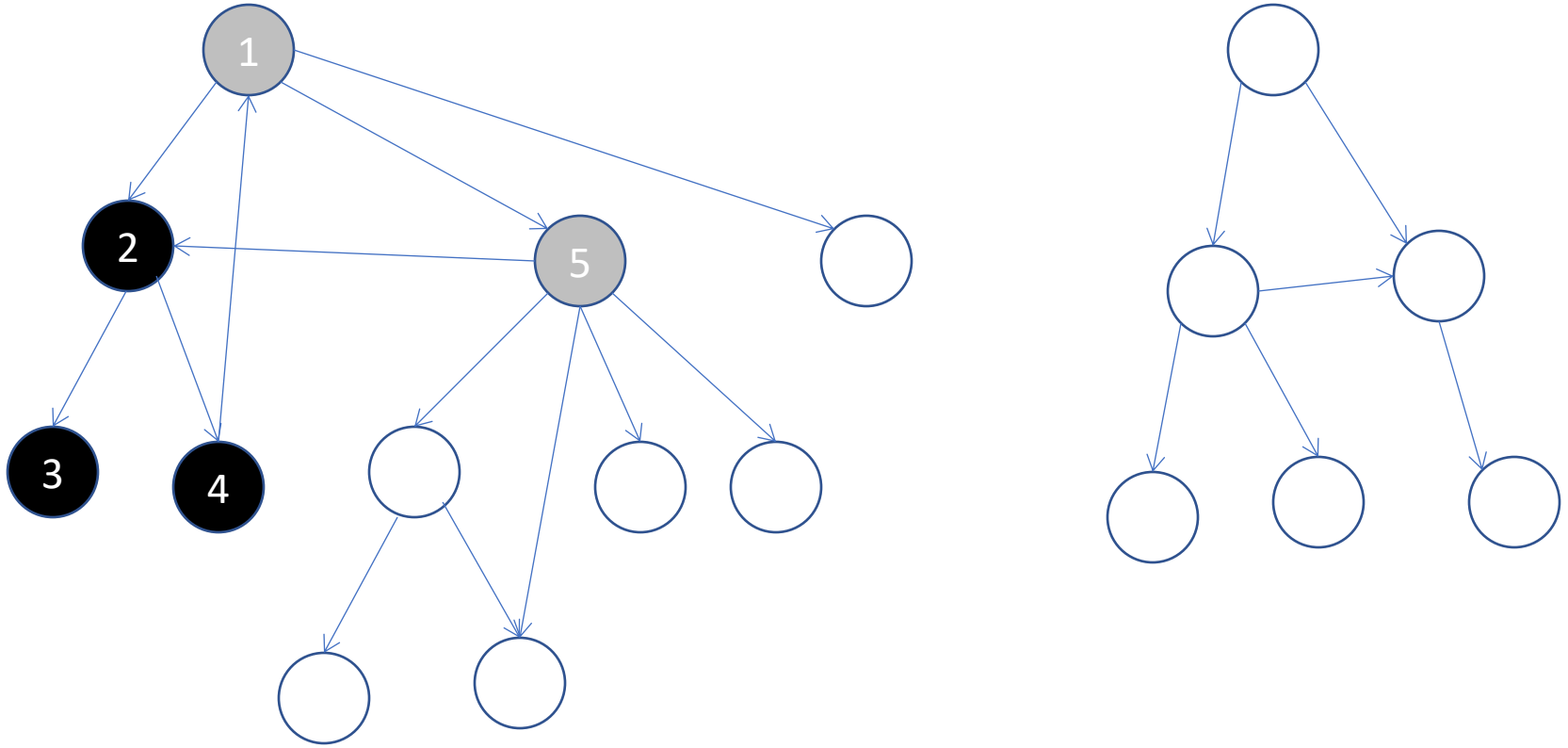
# DFS Traversal



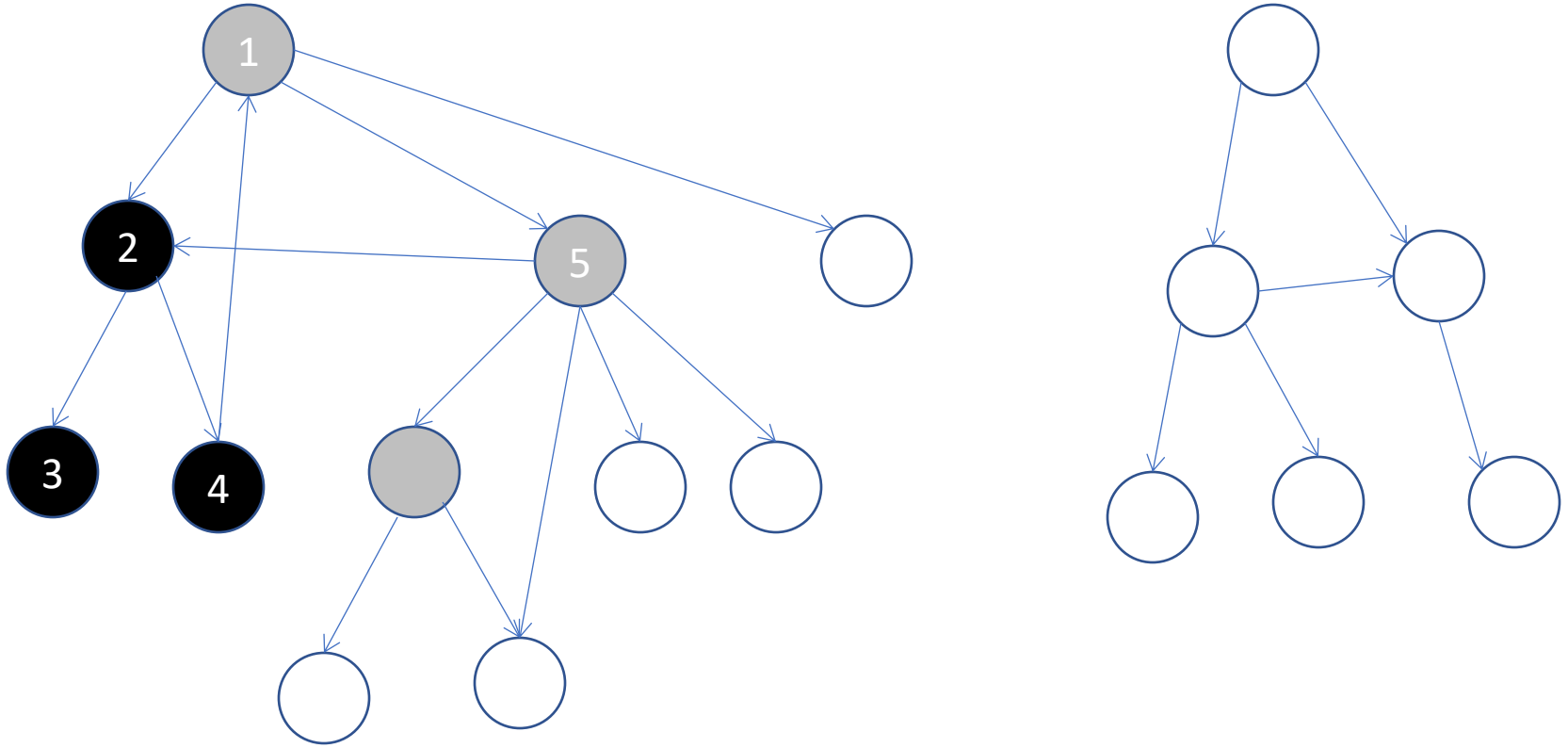
# DFS Traversal



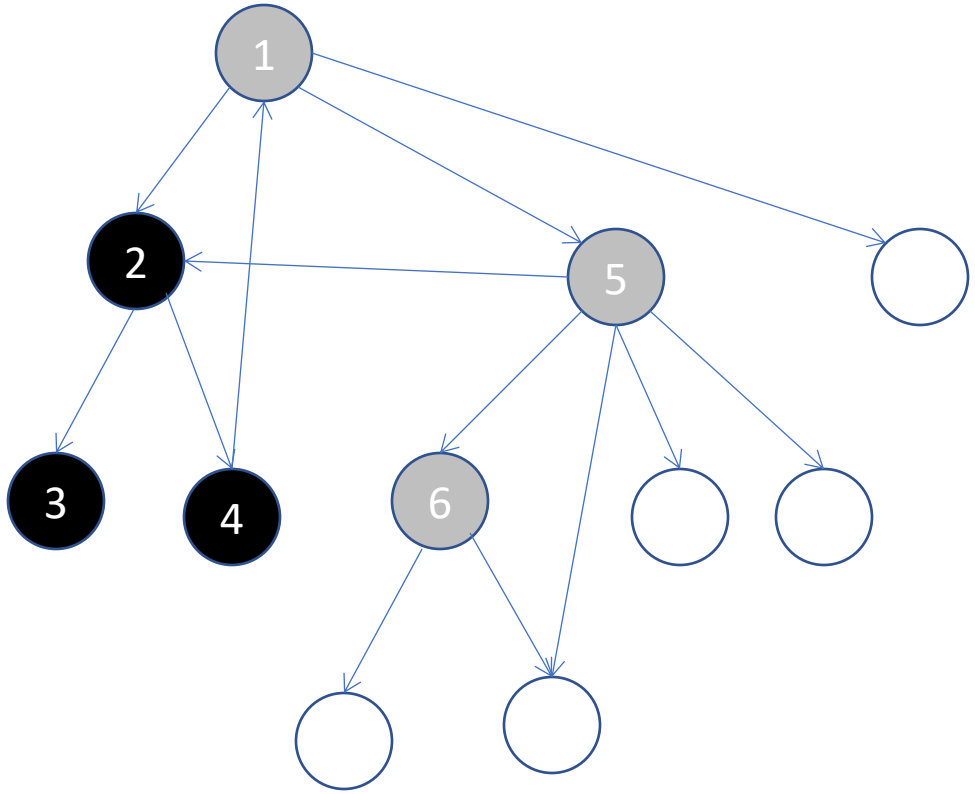
# DFS Traversal

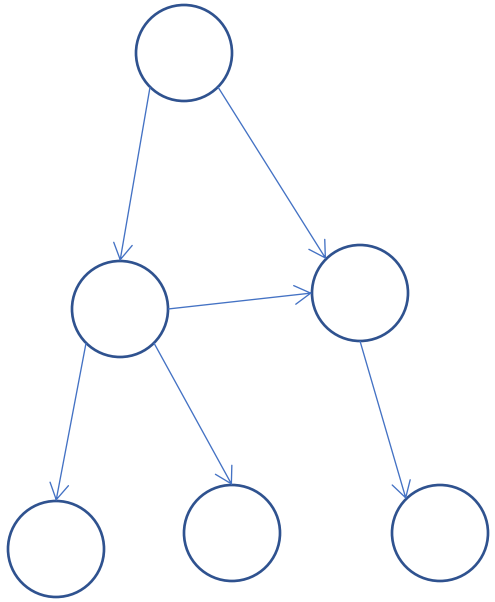


# DFS Traversal

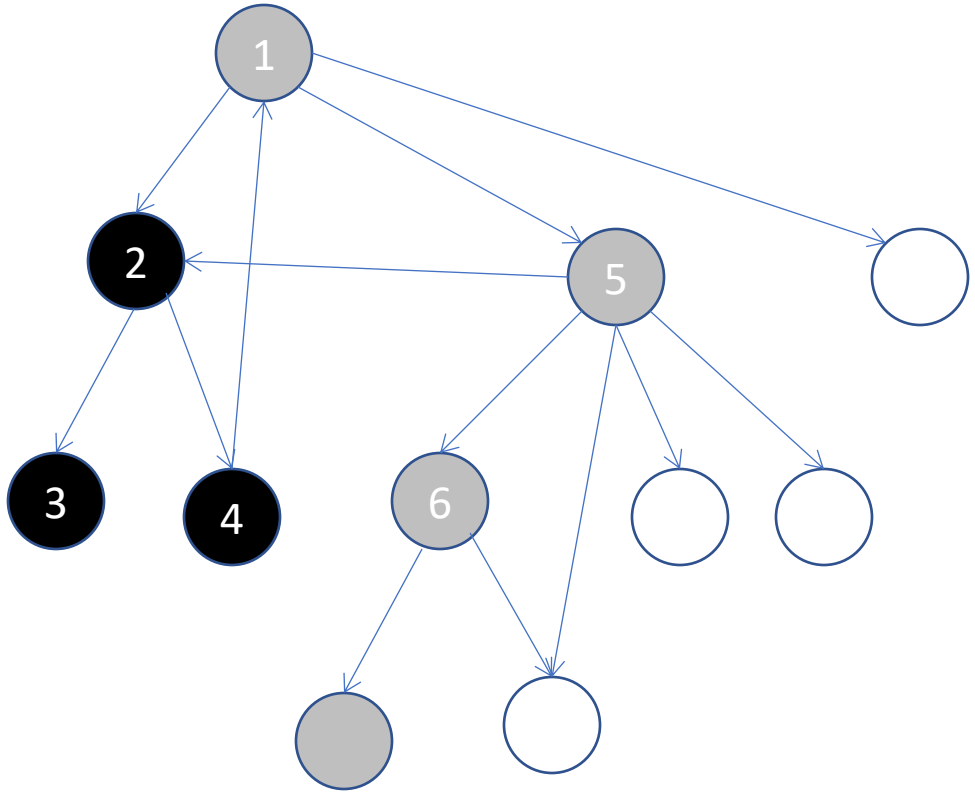


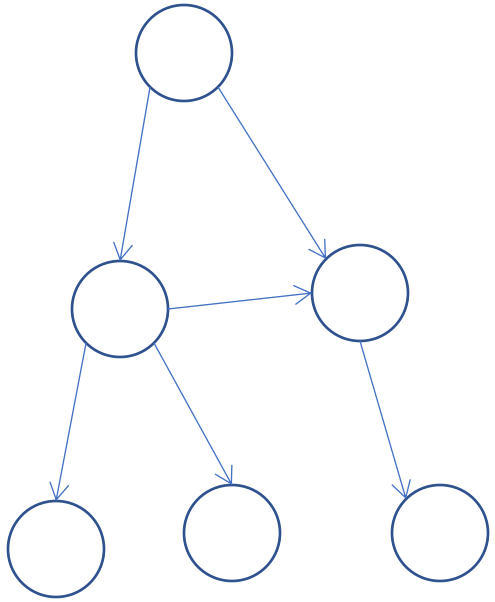
# DFS Traversal



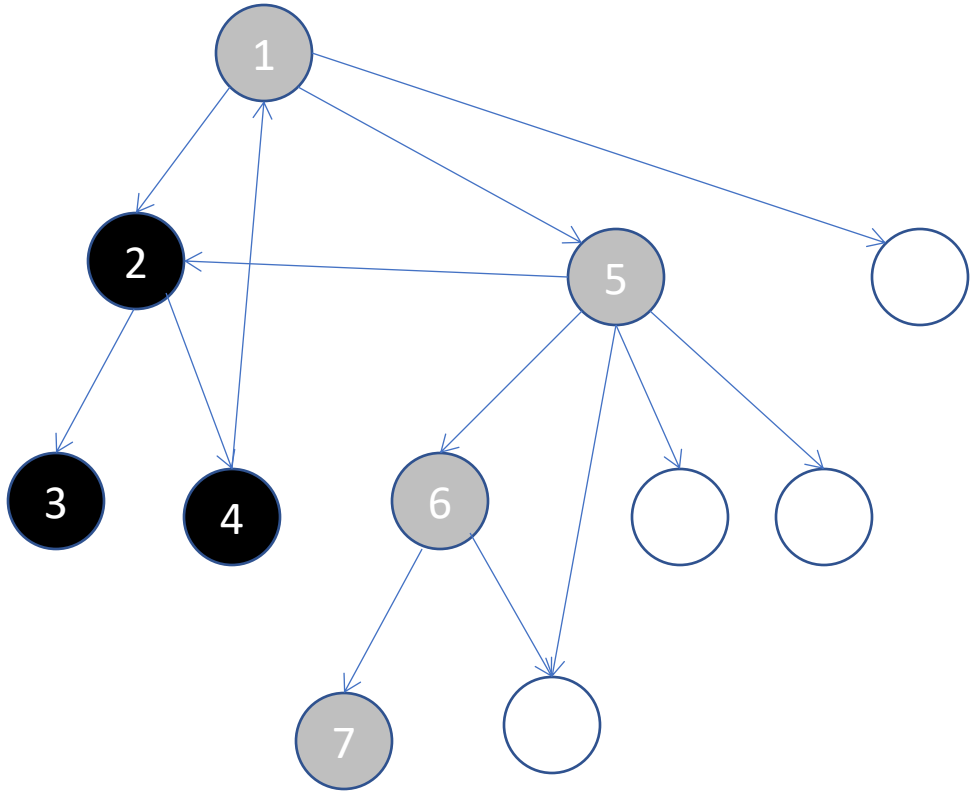



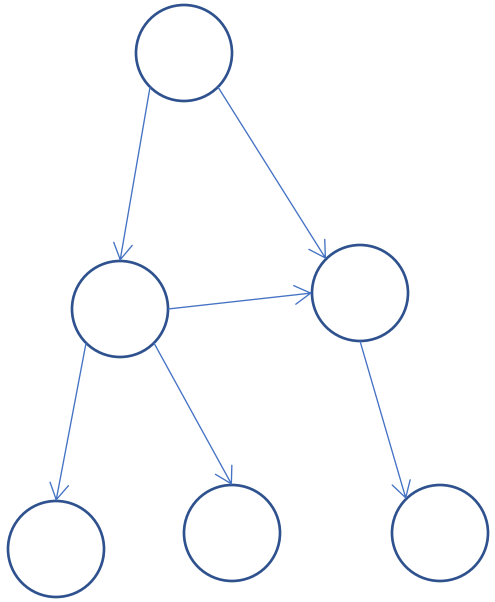
# DFS Traversal



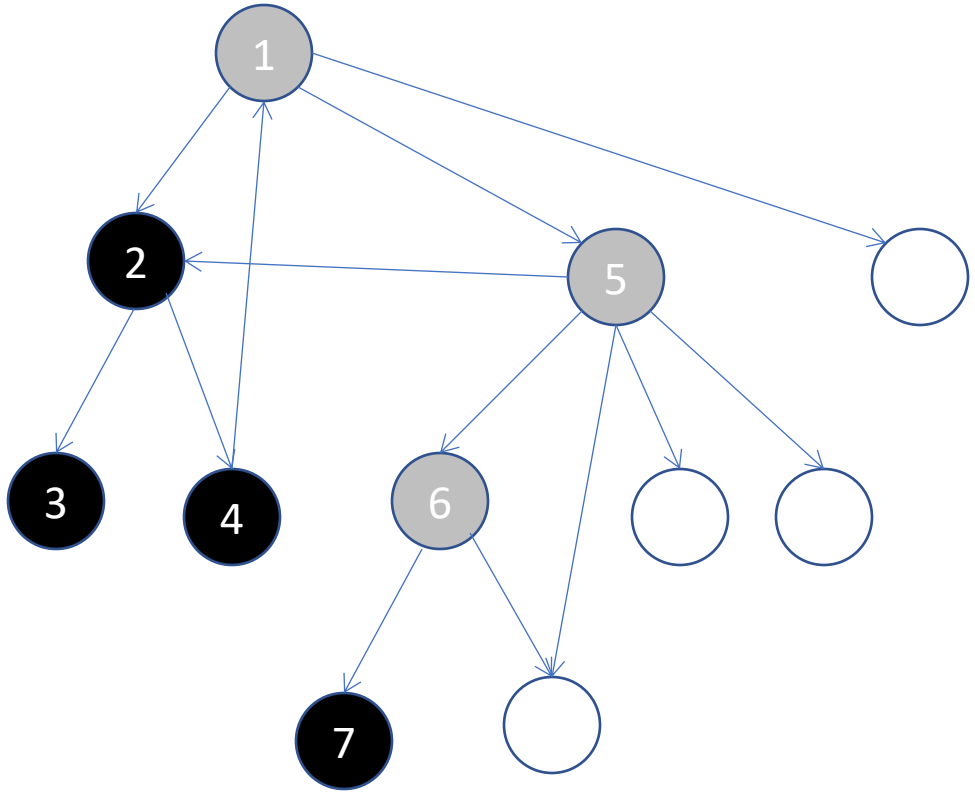



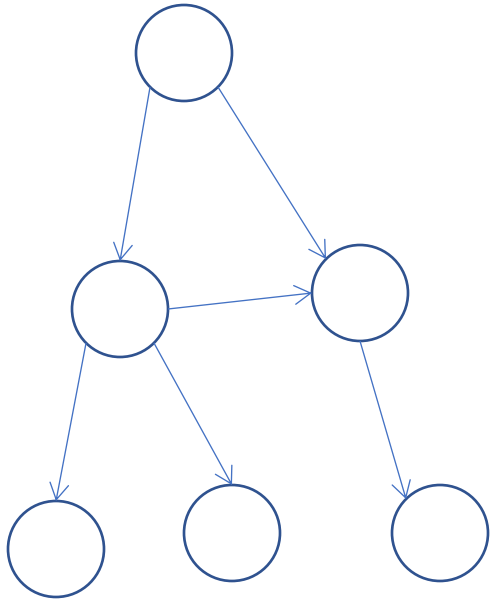
# DFS Traversal



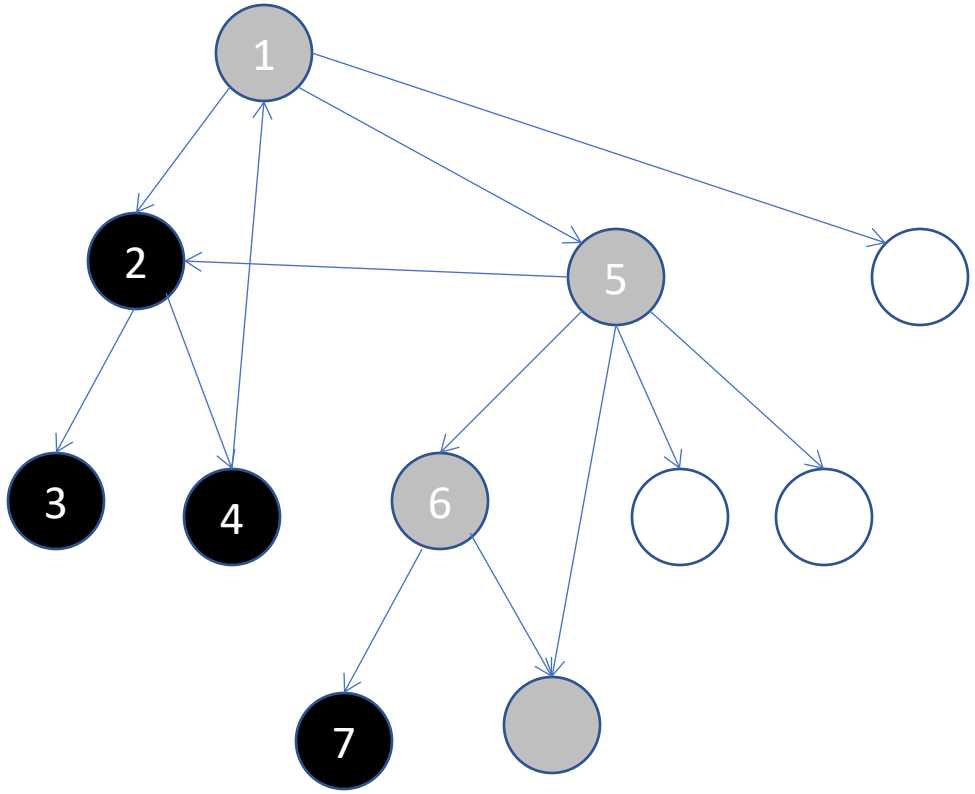



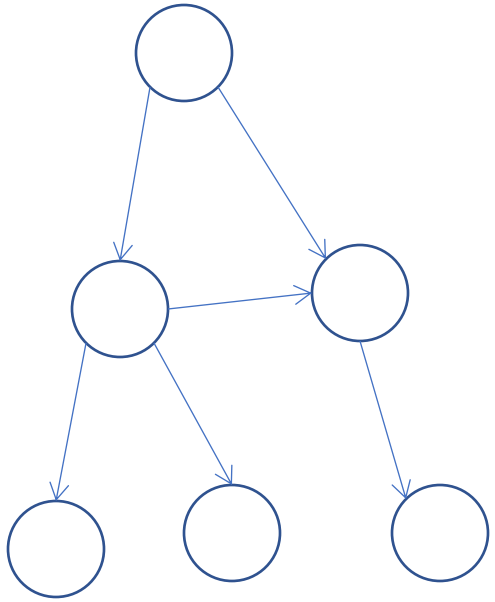
# DFS Traversal



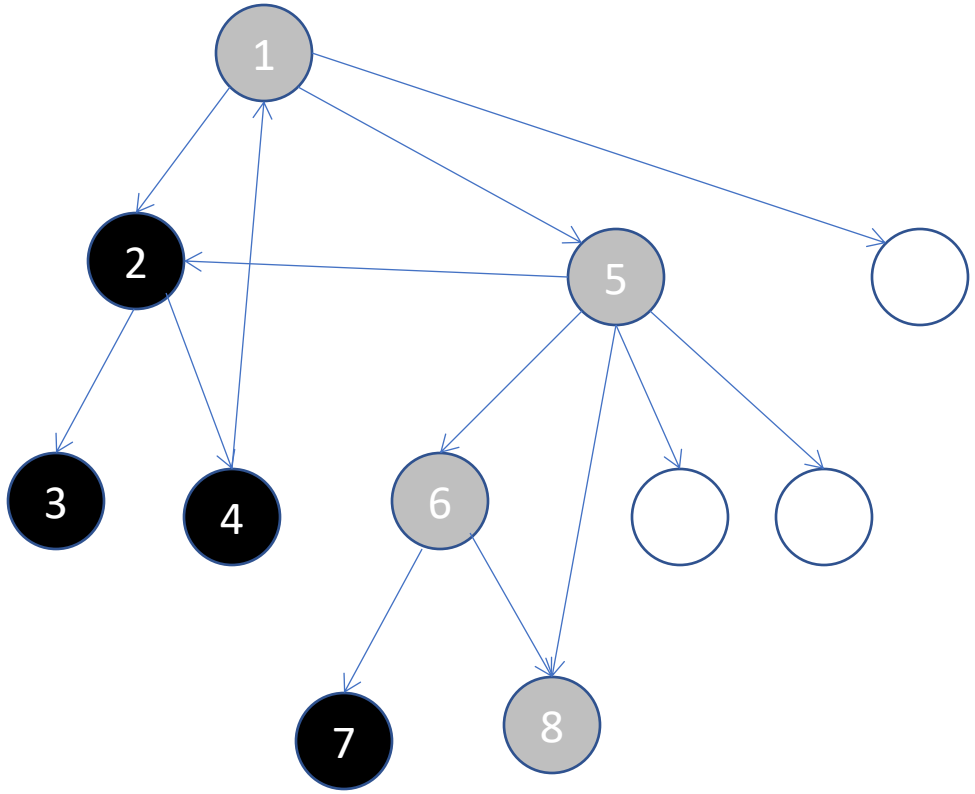



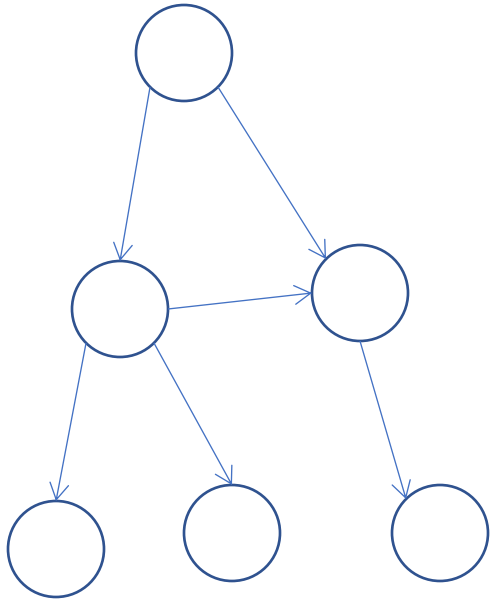
# DFS Traversal



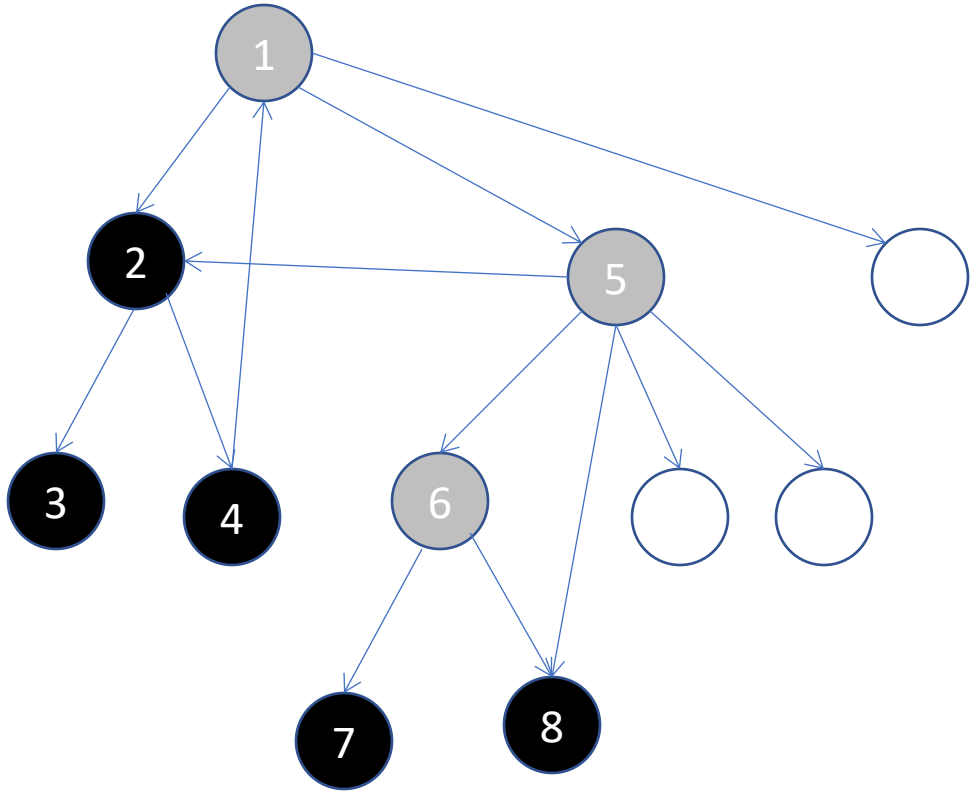



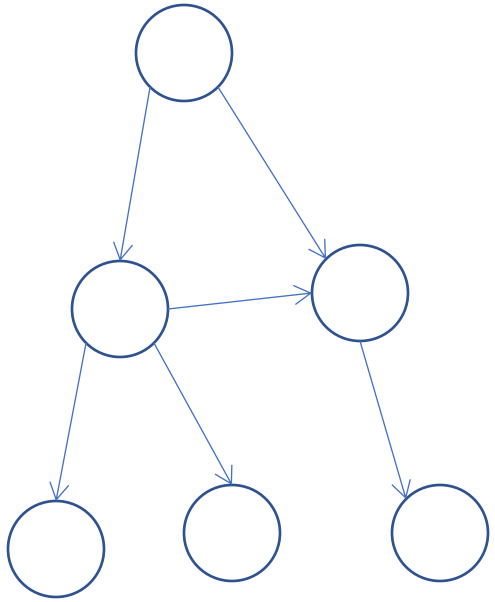
# DFS Traversal



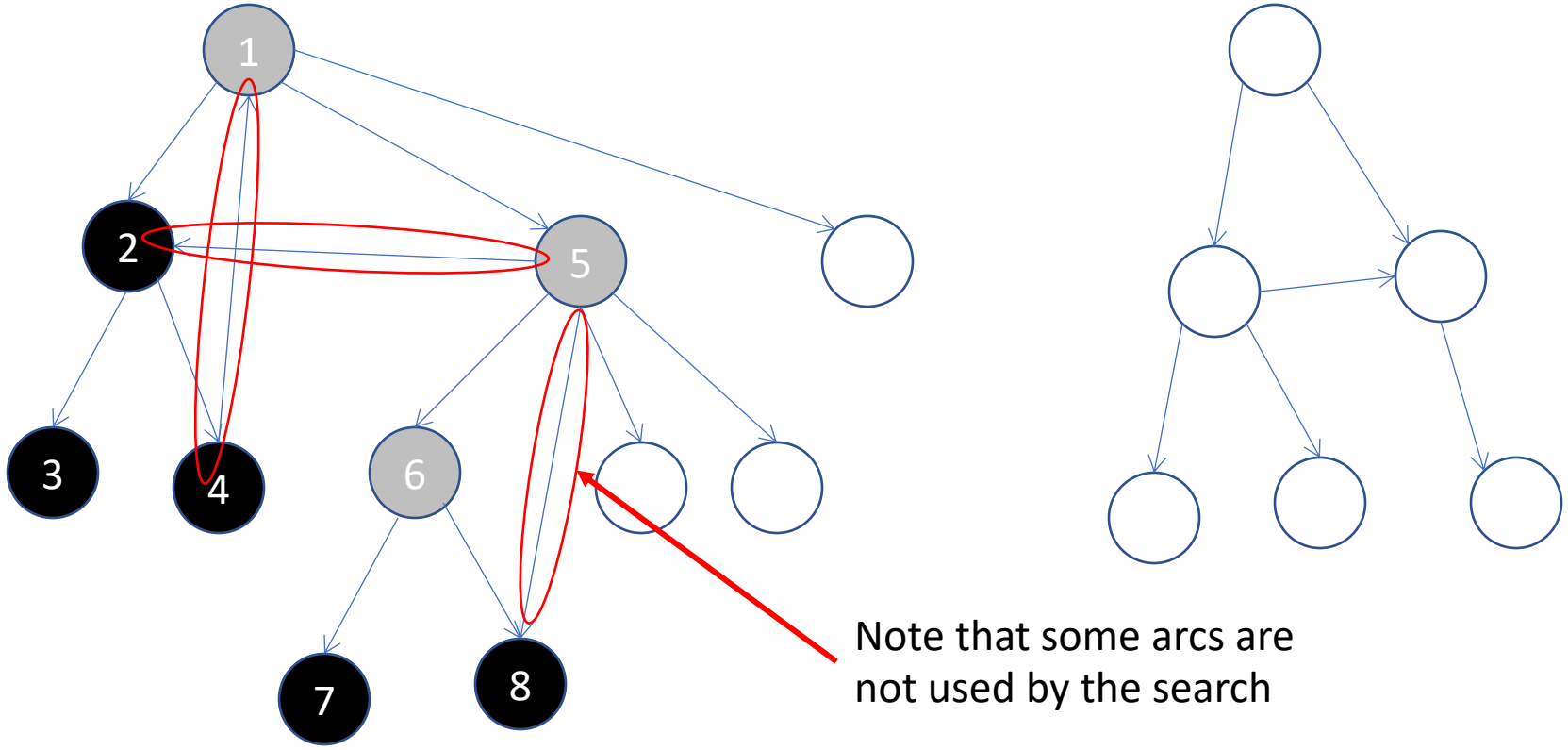



# DFS Traversal

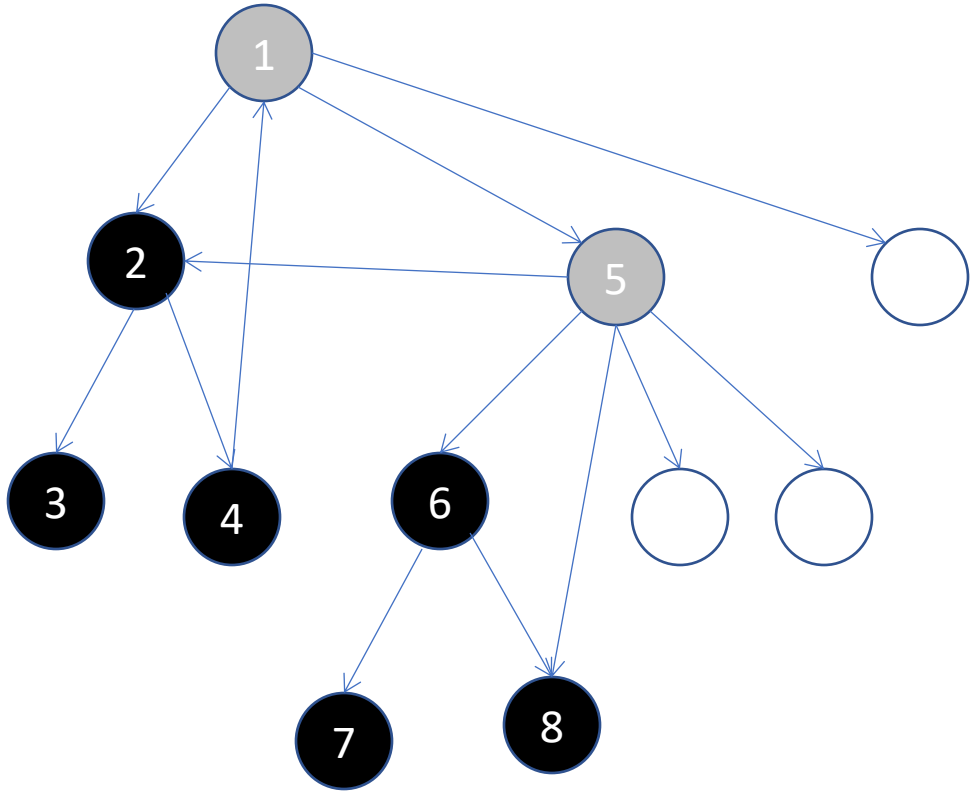


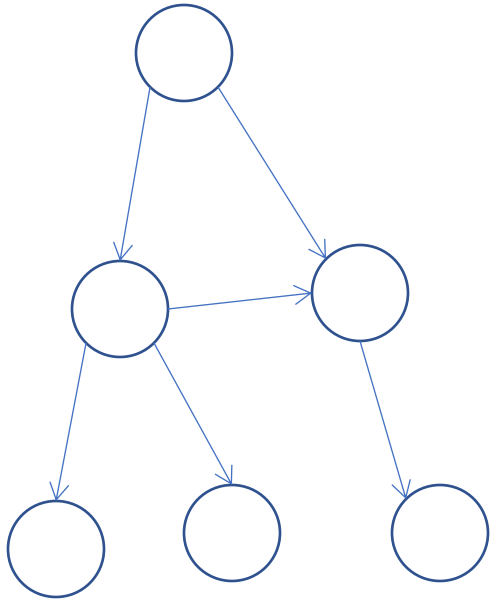



# DFS Traversal

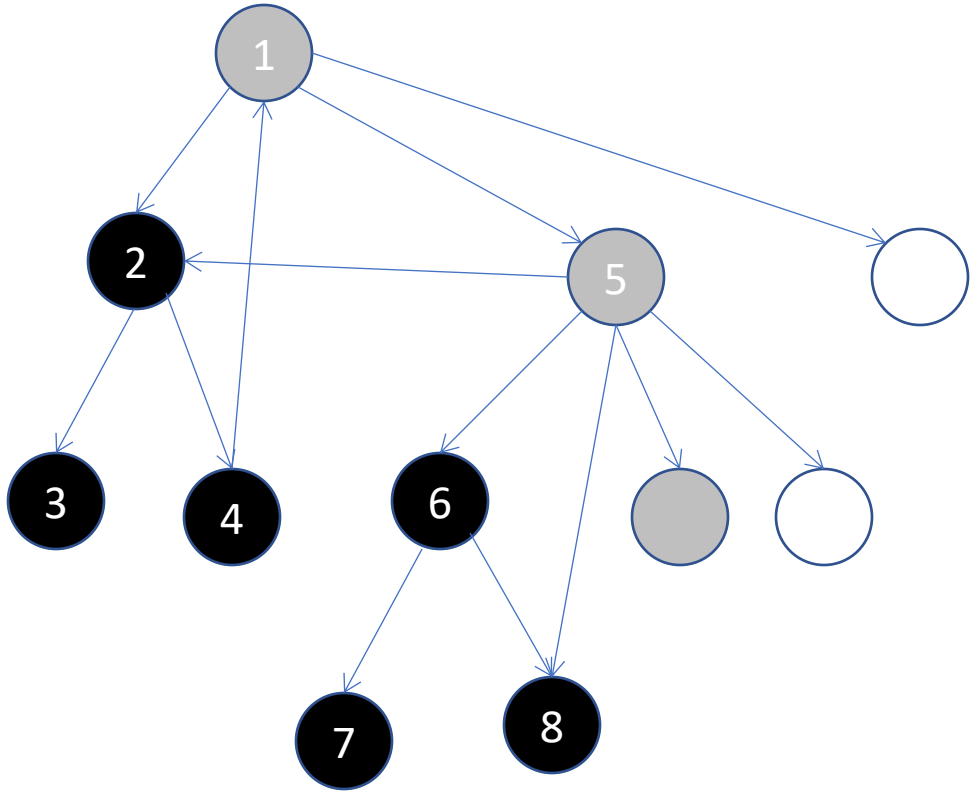


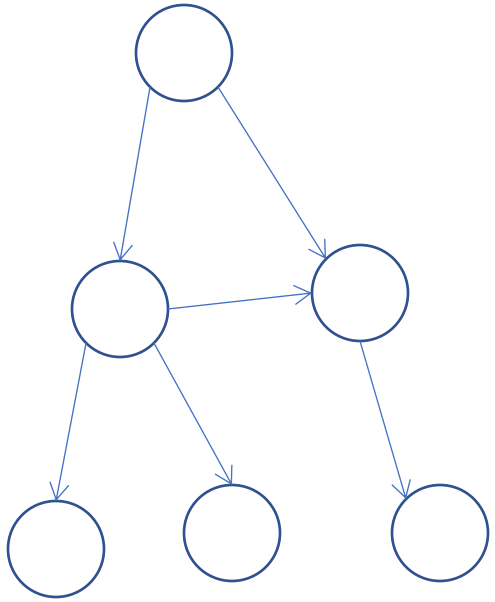
# DFS Traversal



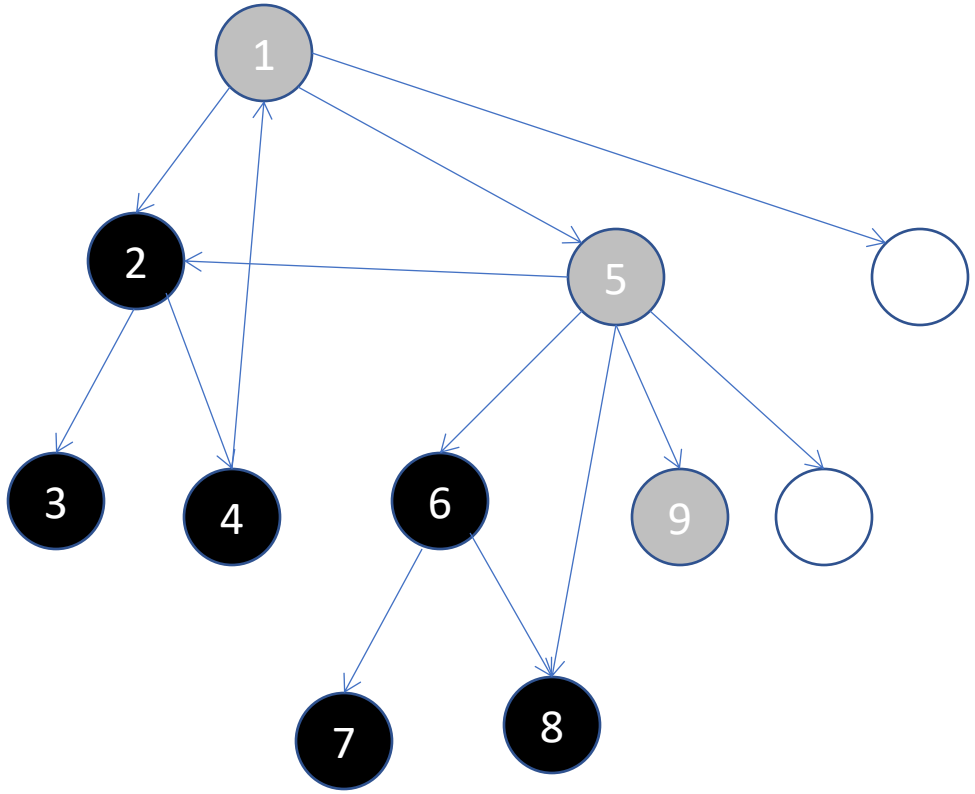



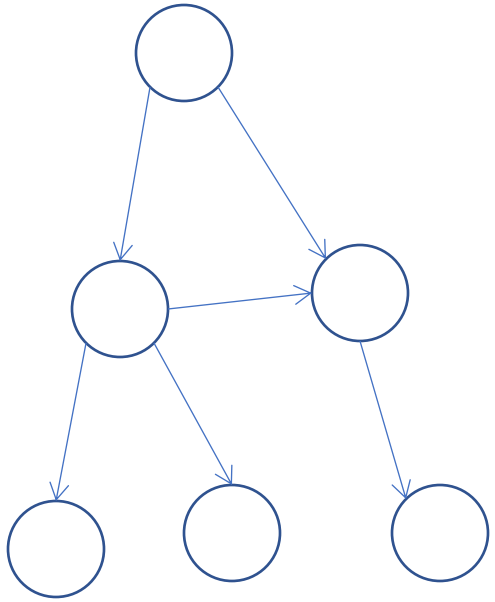
# DFS Traversal



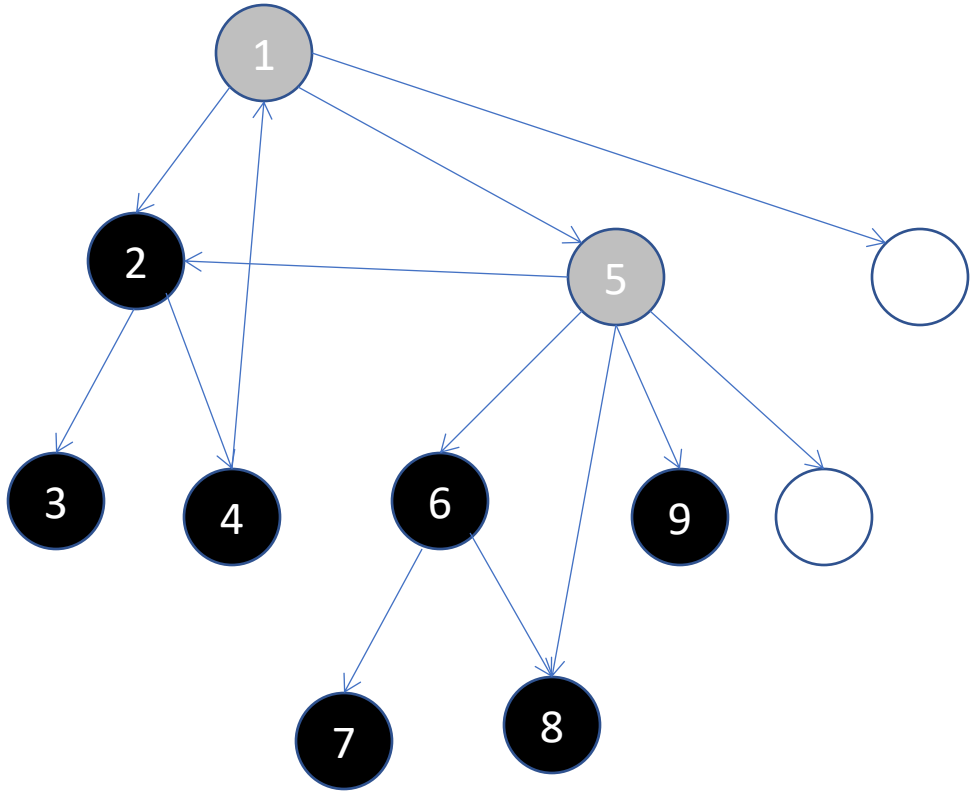



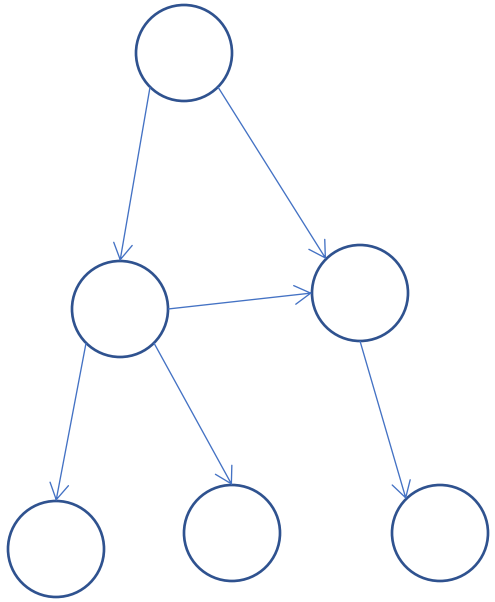
# DFS Traversal



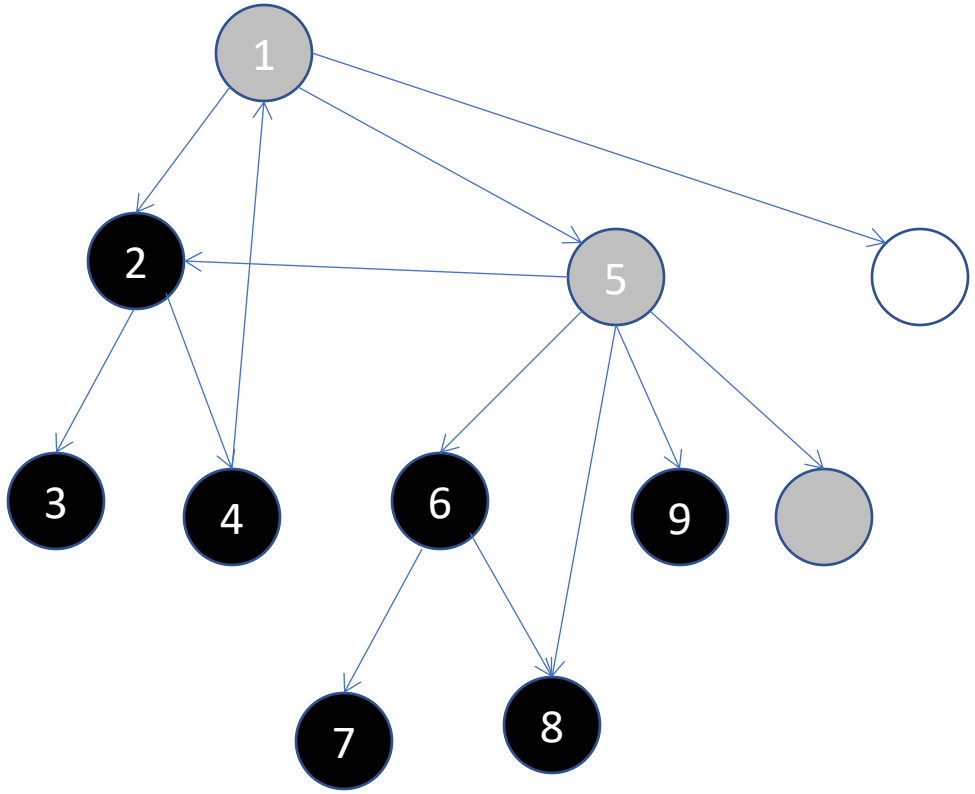



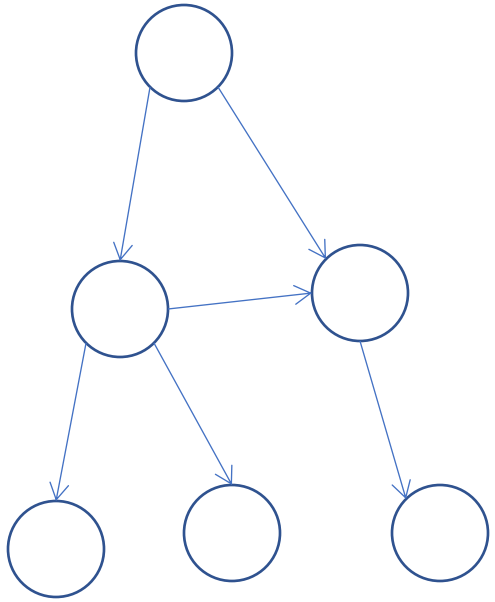
# DFS Traversal



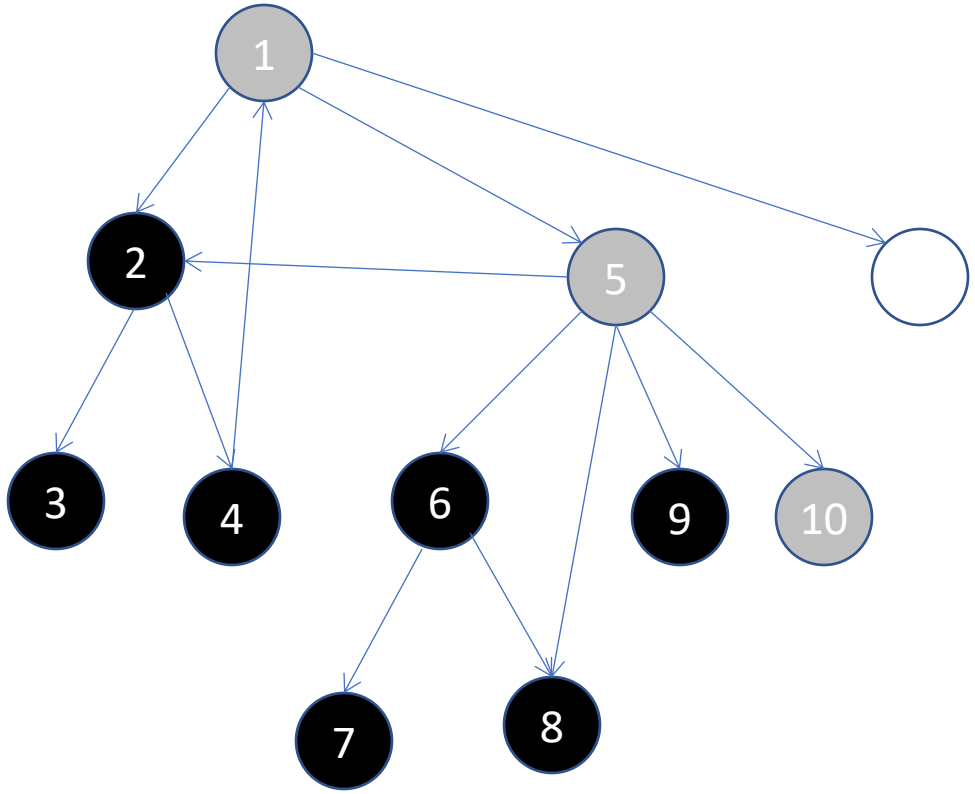



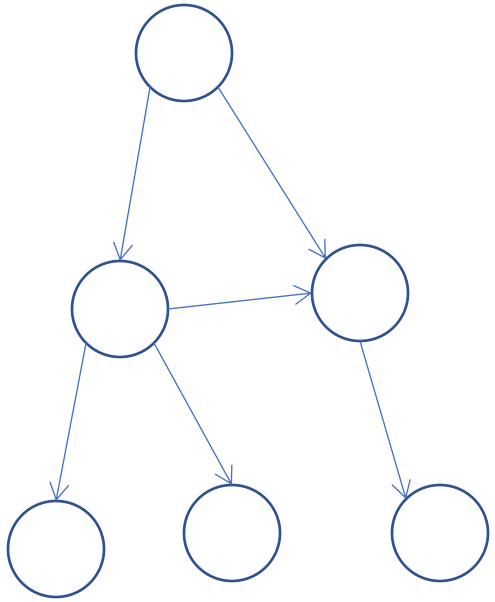
# DFS Traversal



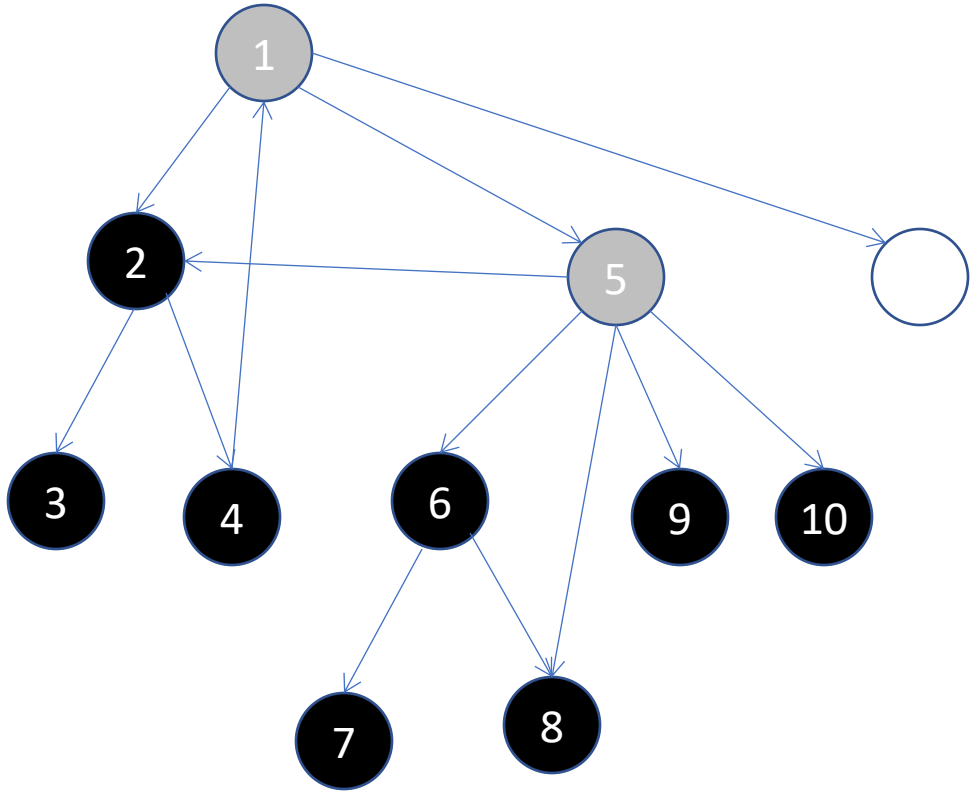



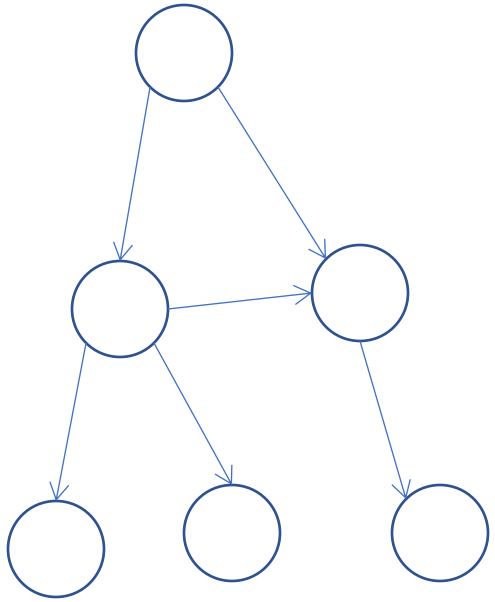
# DFS Traversal



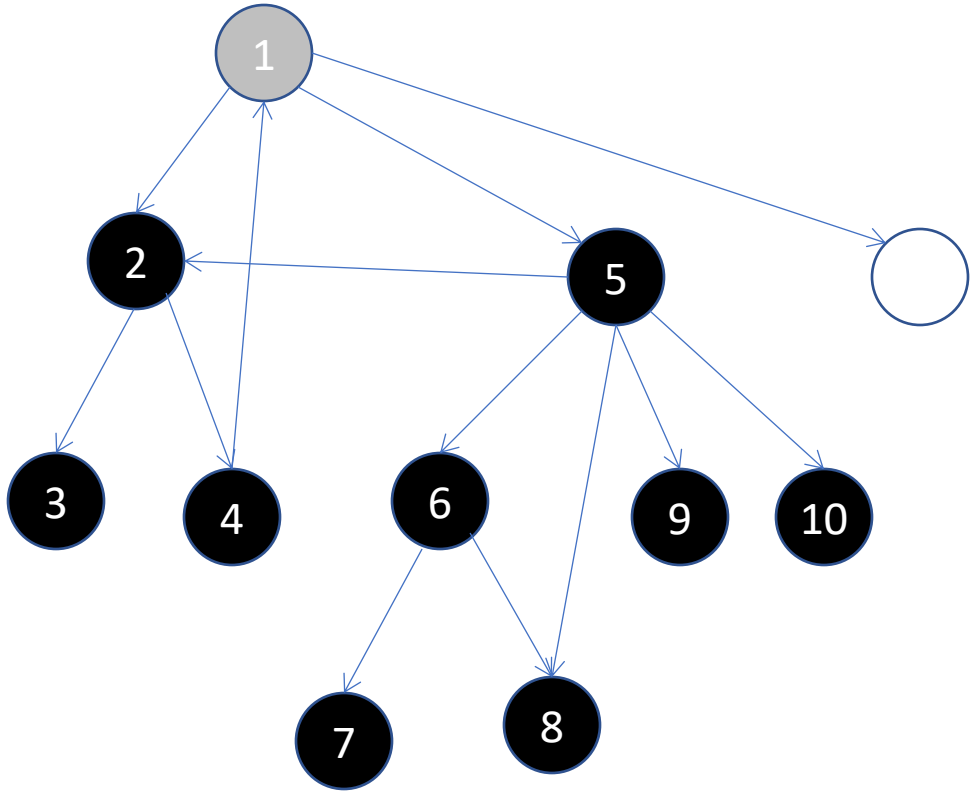



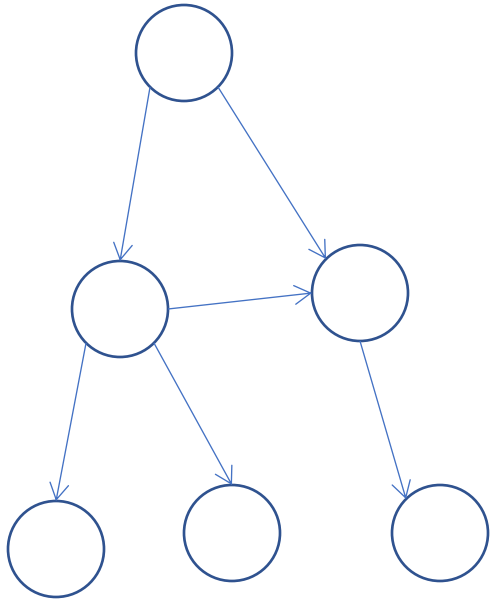
# DFS Traversal



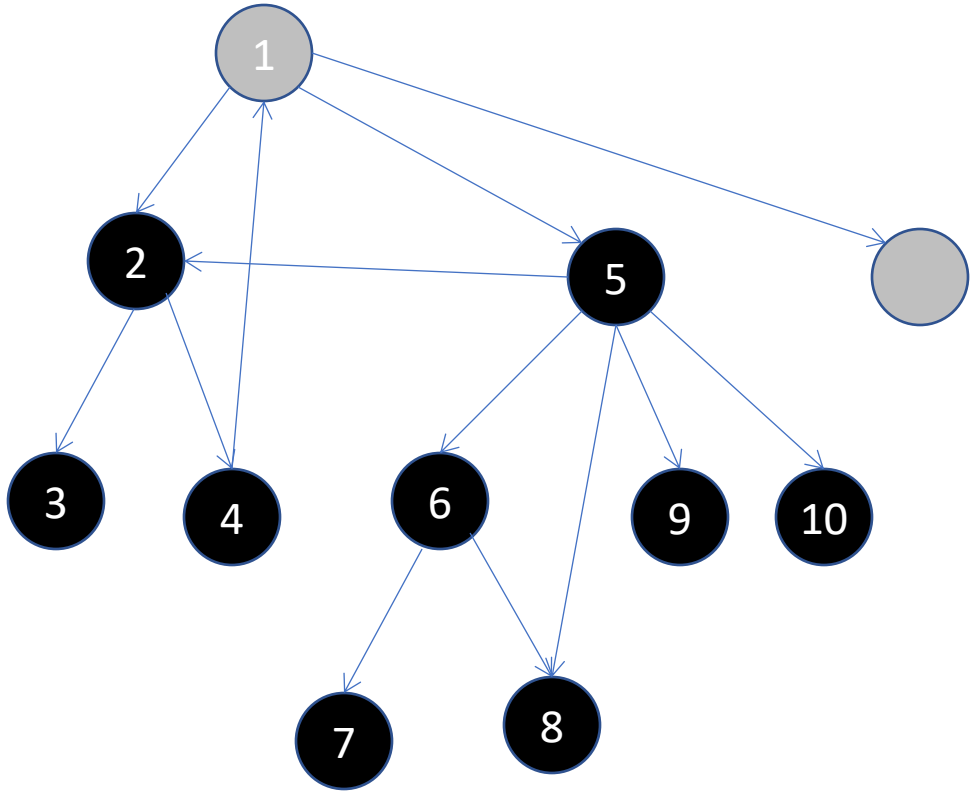



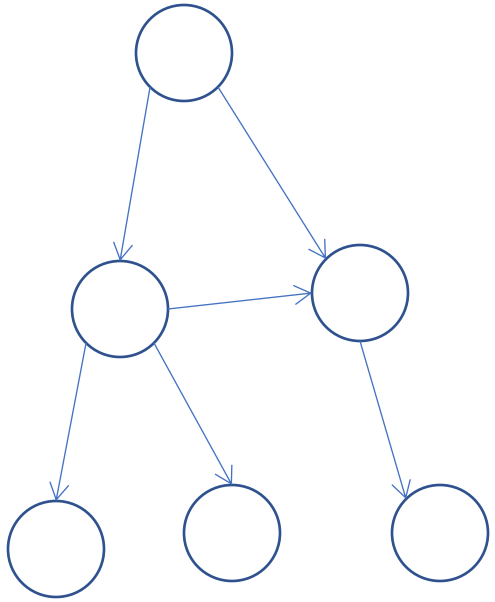
# DFS Traversal



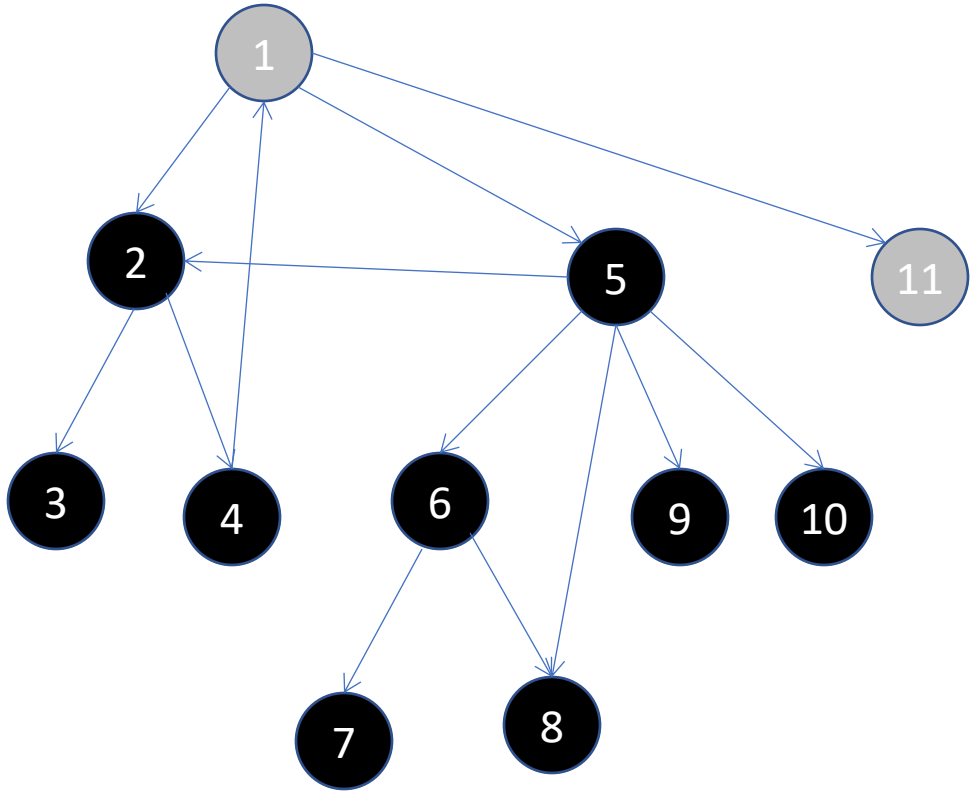



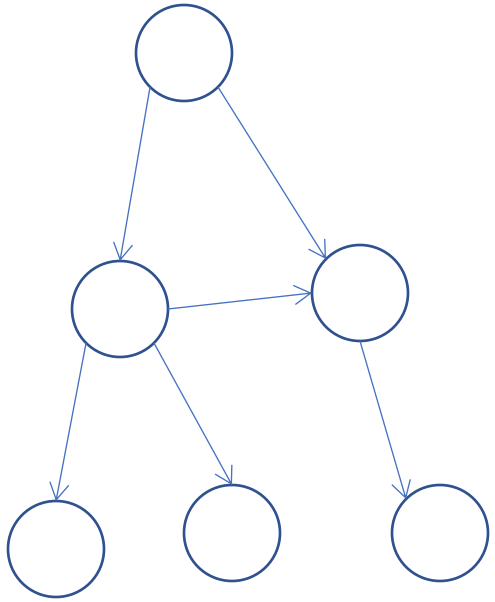
# DFS Traversal



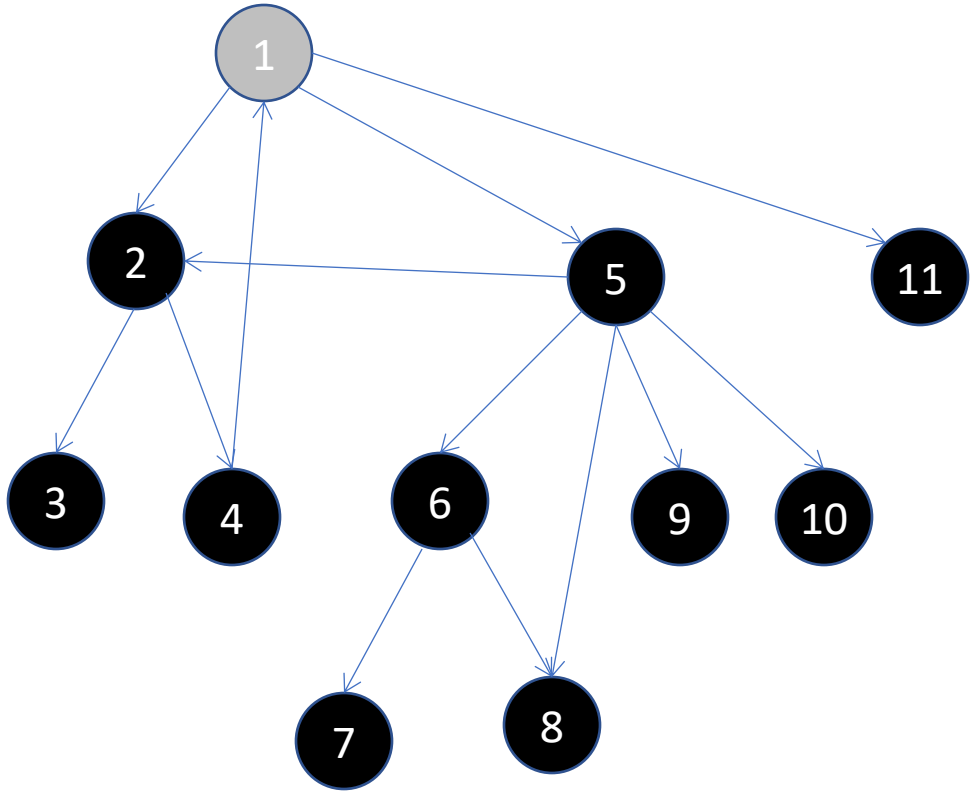



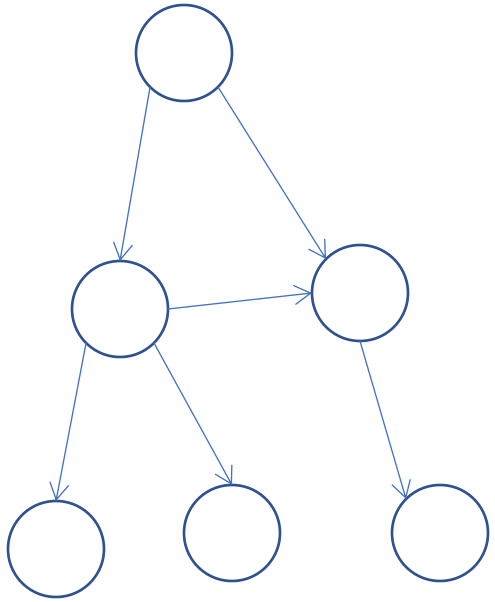
# DFS Traversal



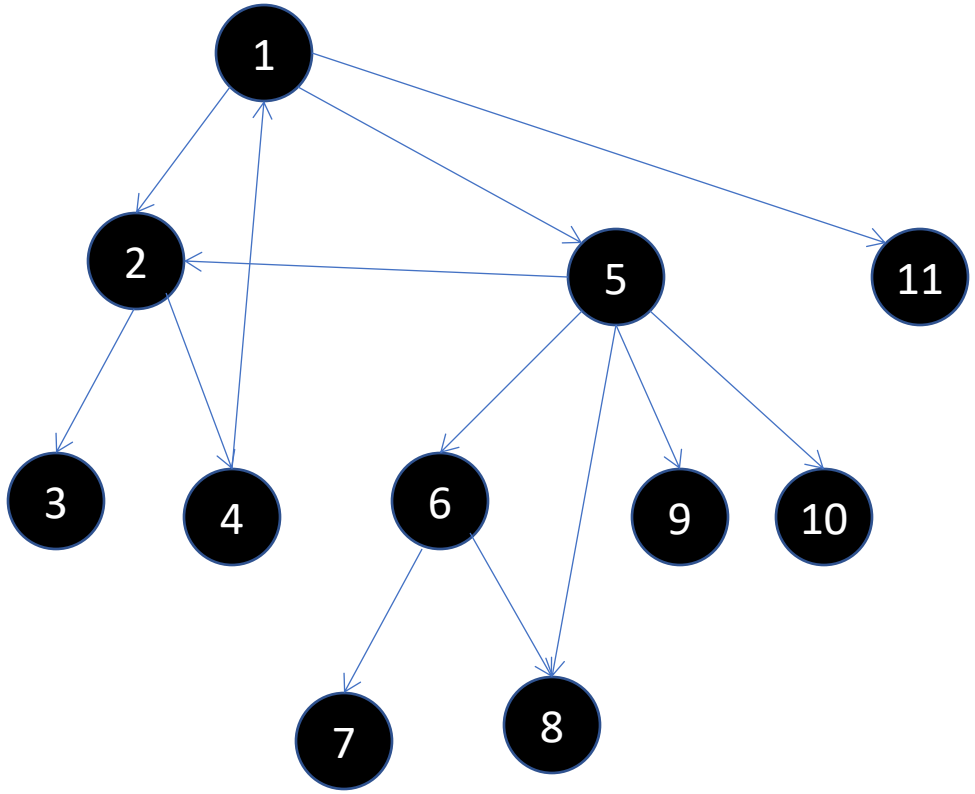



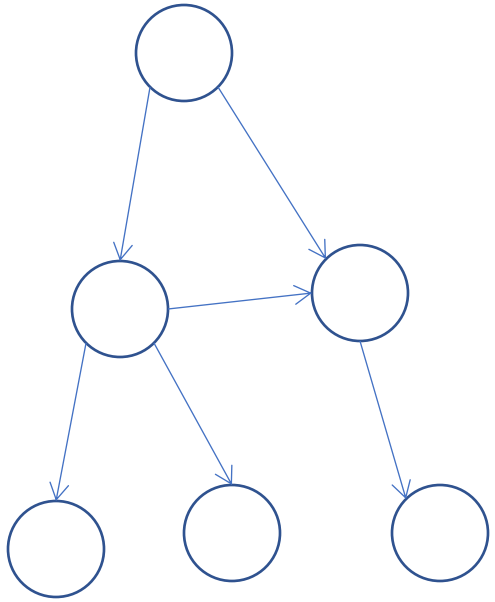
# DFS Traversal



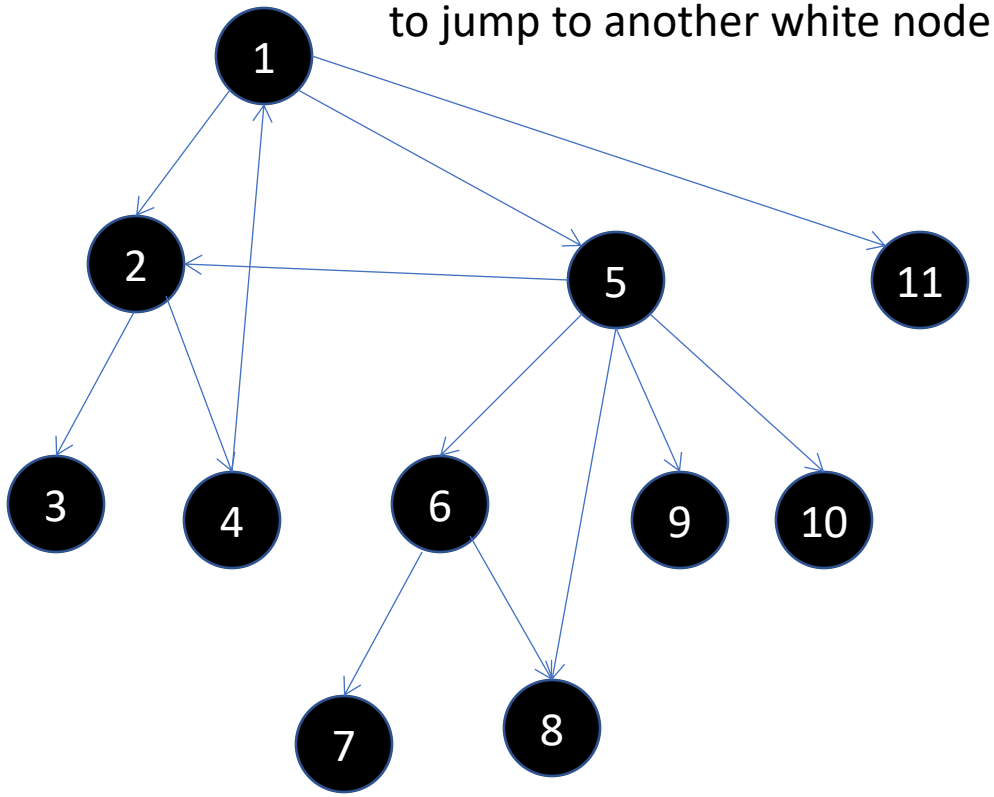
# DFS Traversal

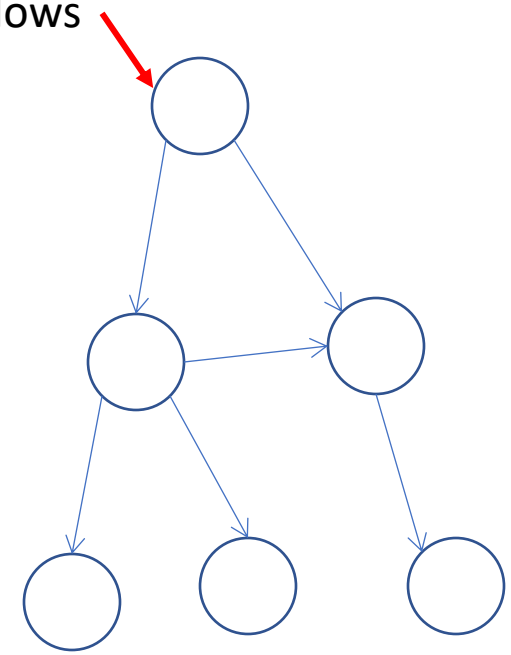


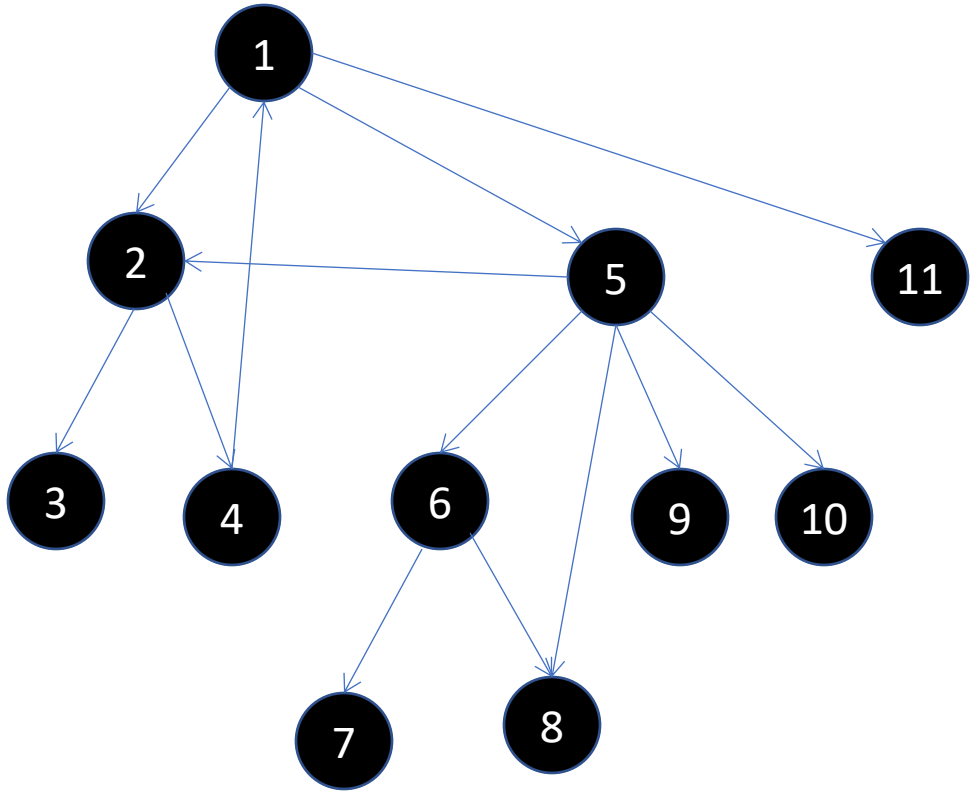
# DFS Traversal

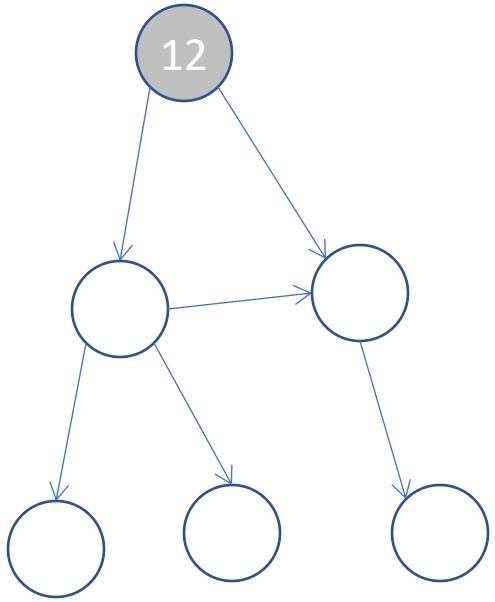
Now we are done with the inner loop starting from node 1. The outer loop allows to jump to another white node



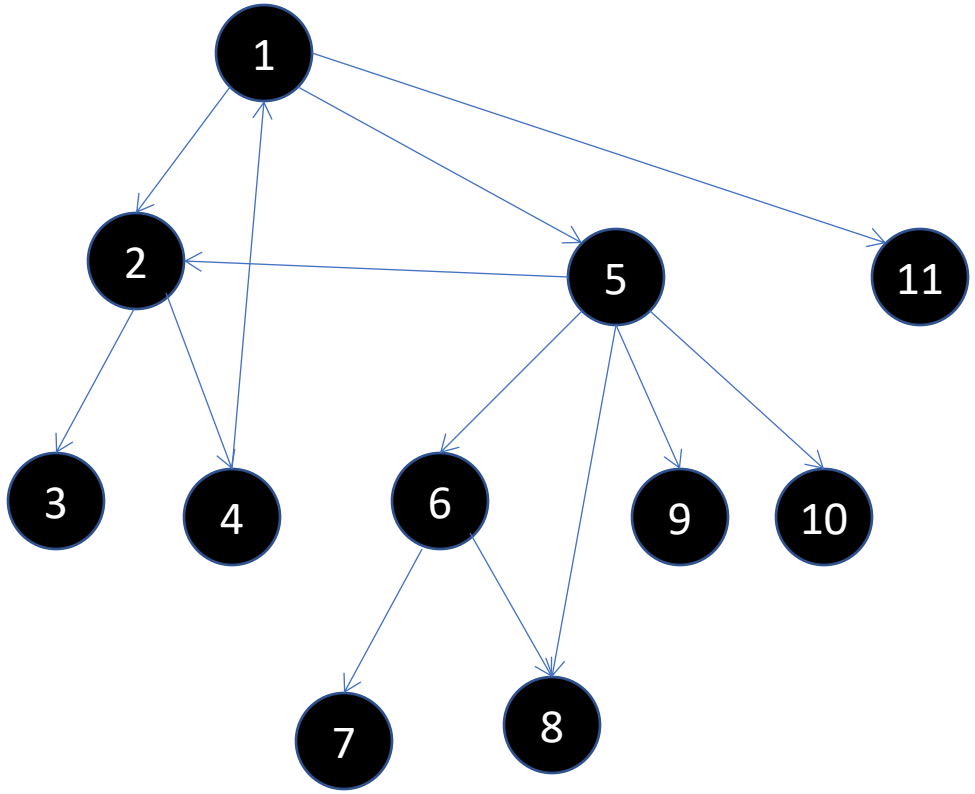



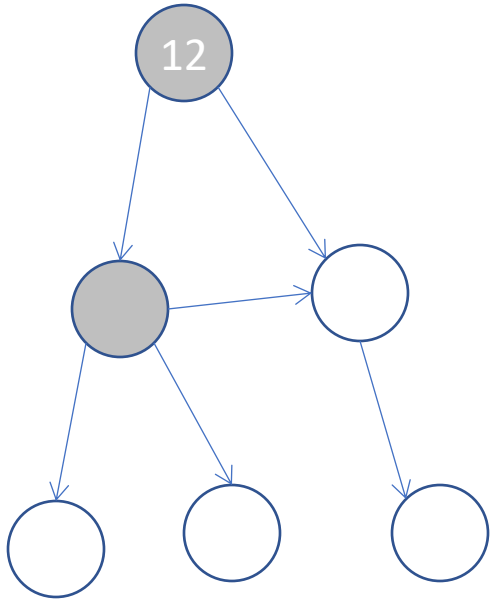
# DFS Traversal



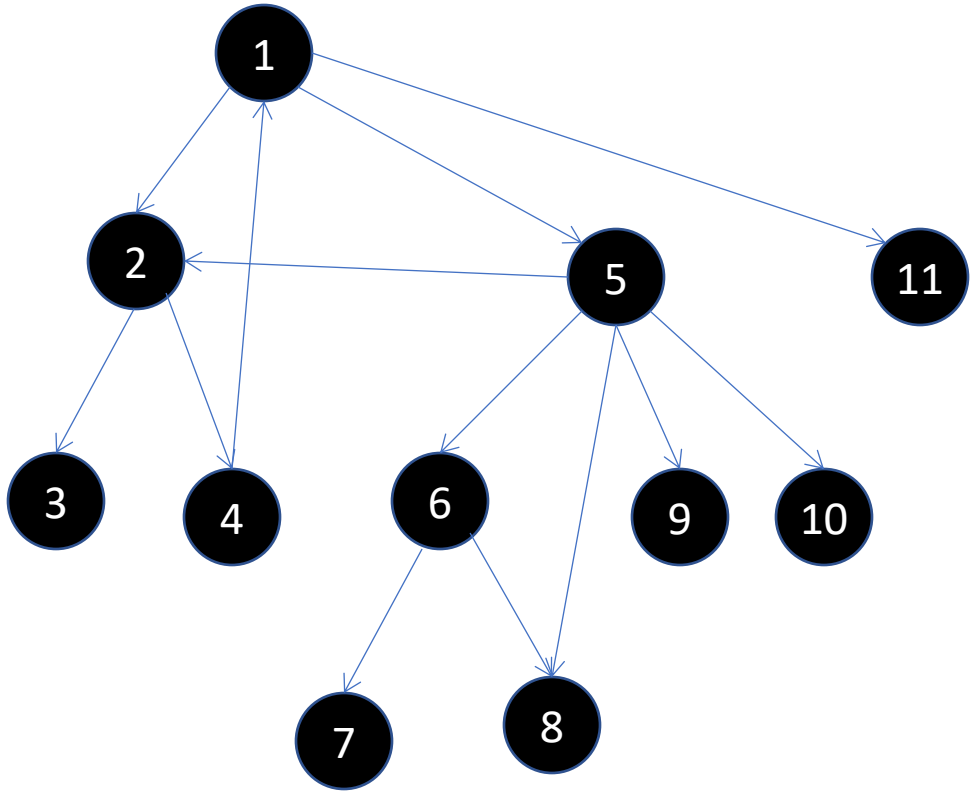



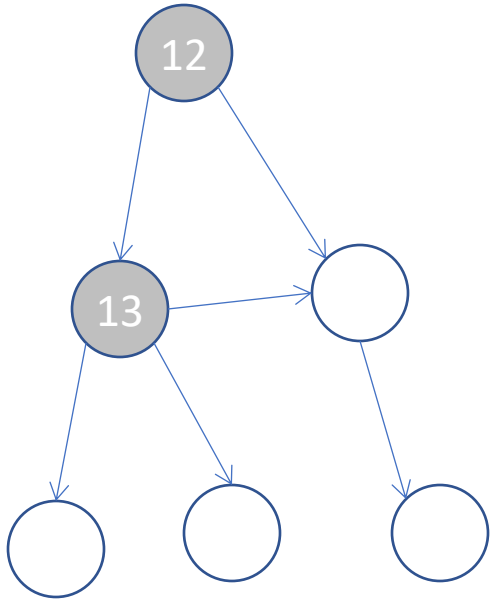
# DFS Traversal



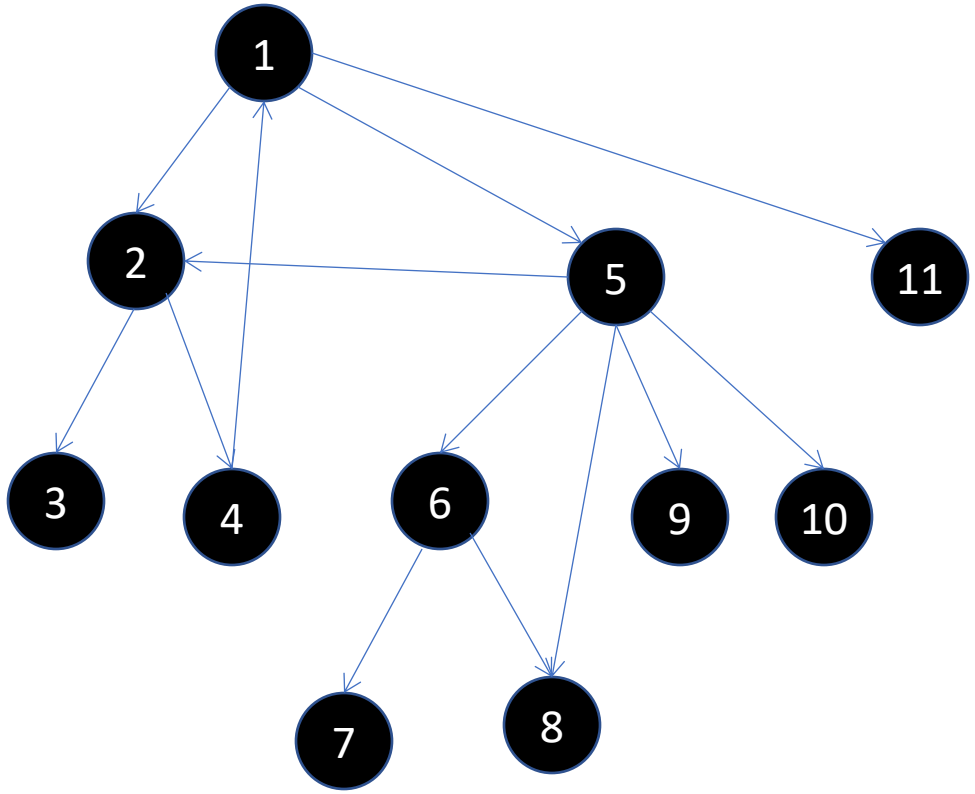



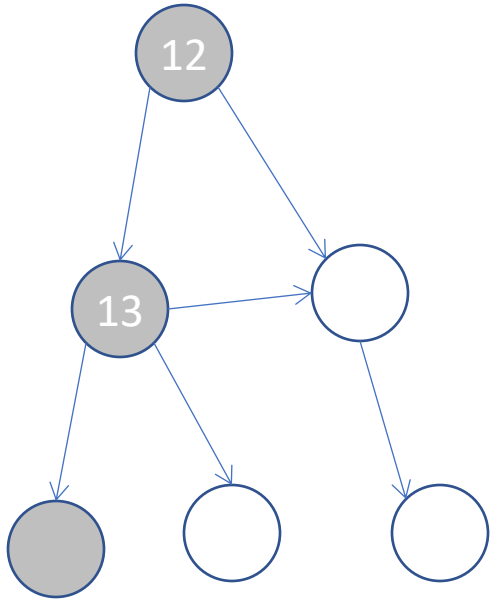
# DFS Traversal



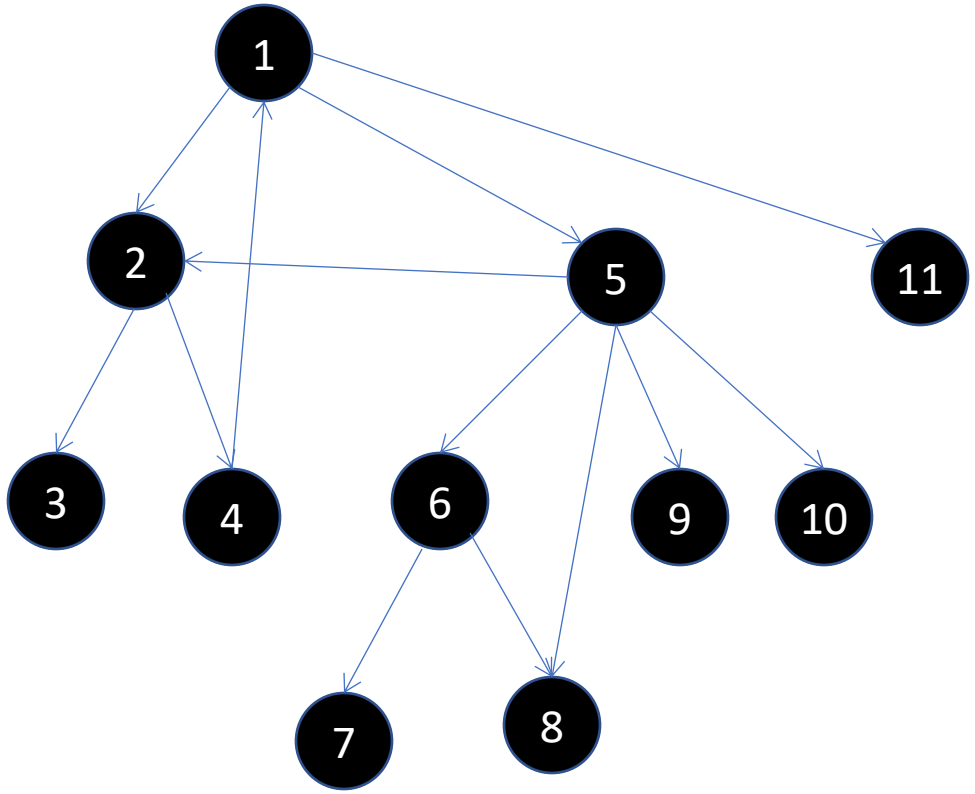



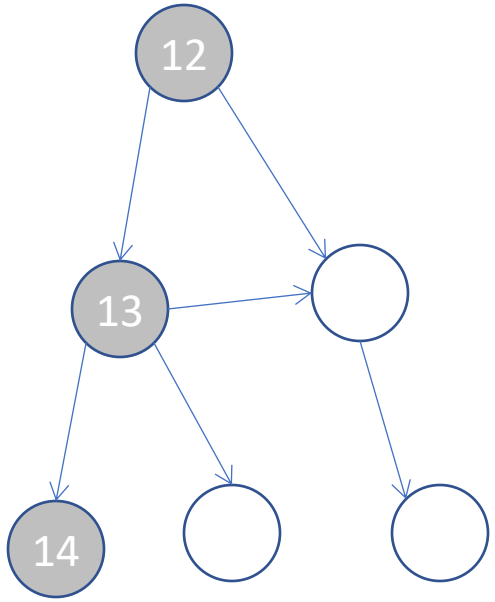
# DFS Traversal



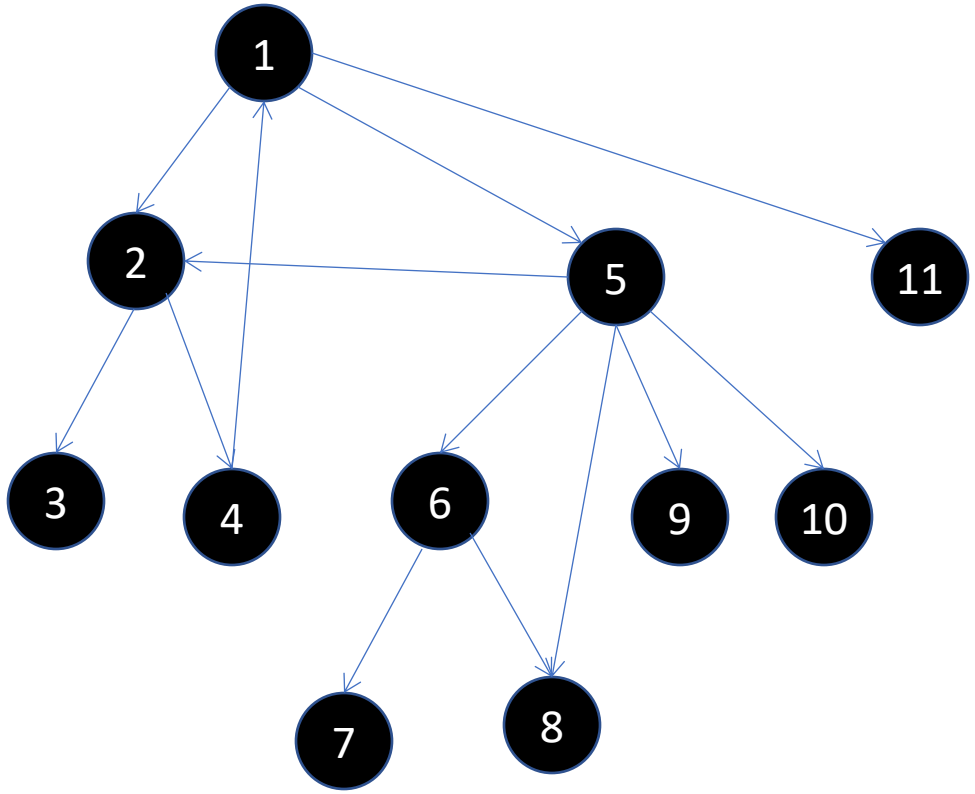



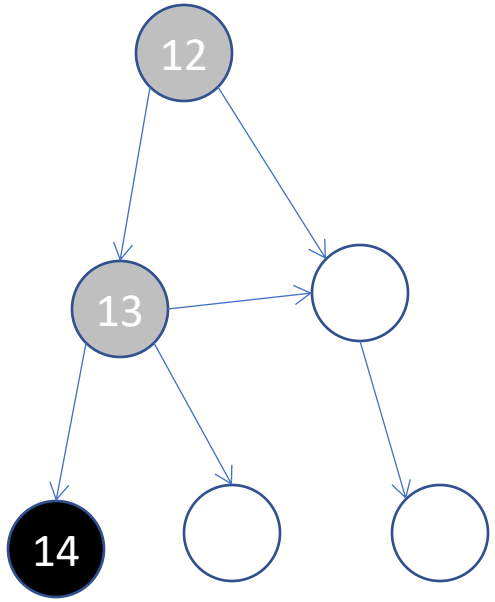
# DFS Traversal



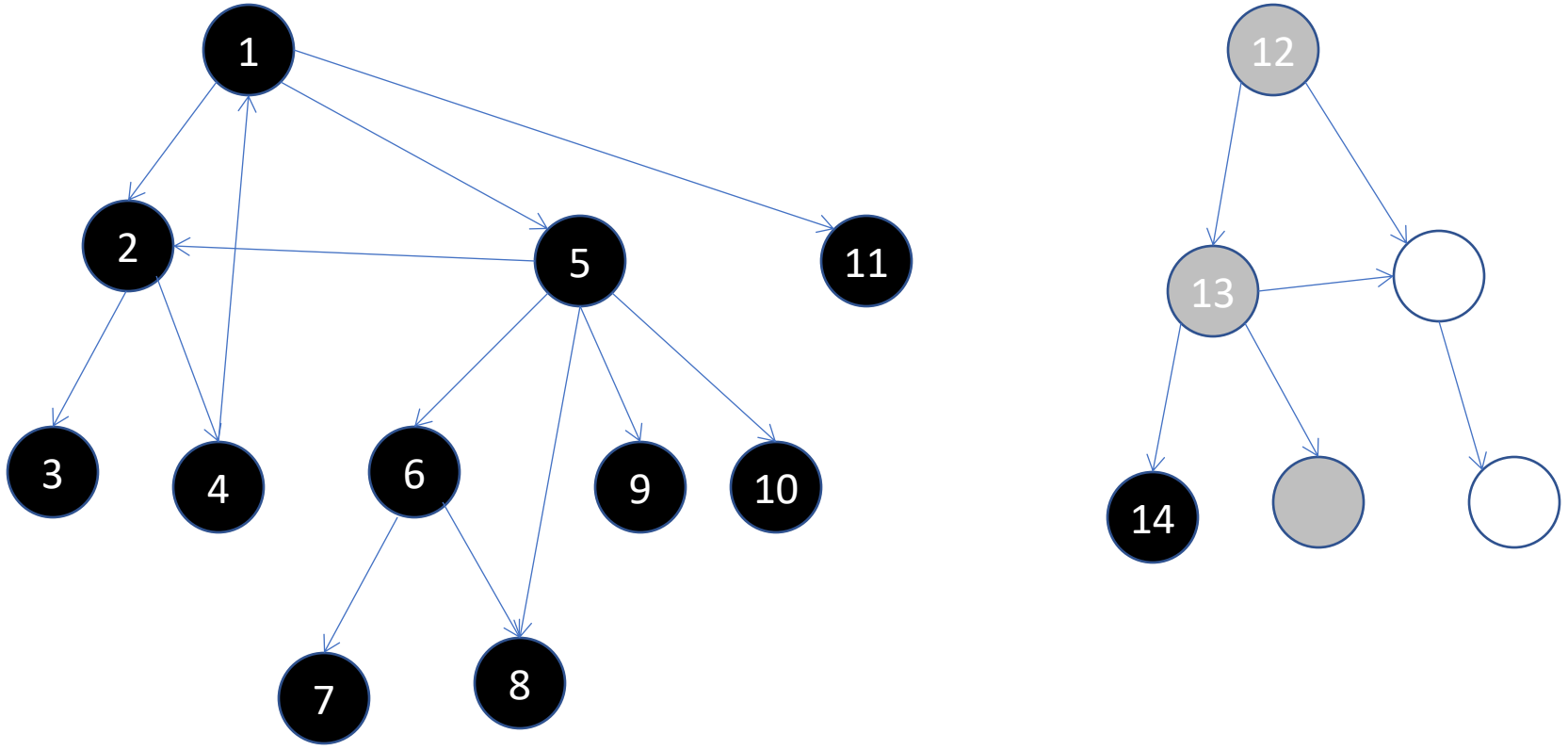



# DFS Traversal

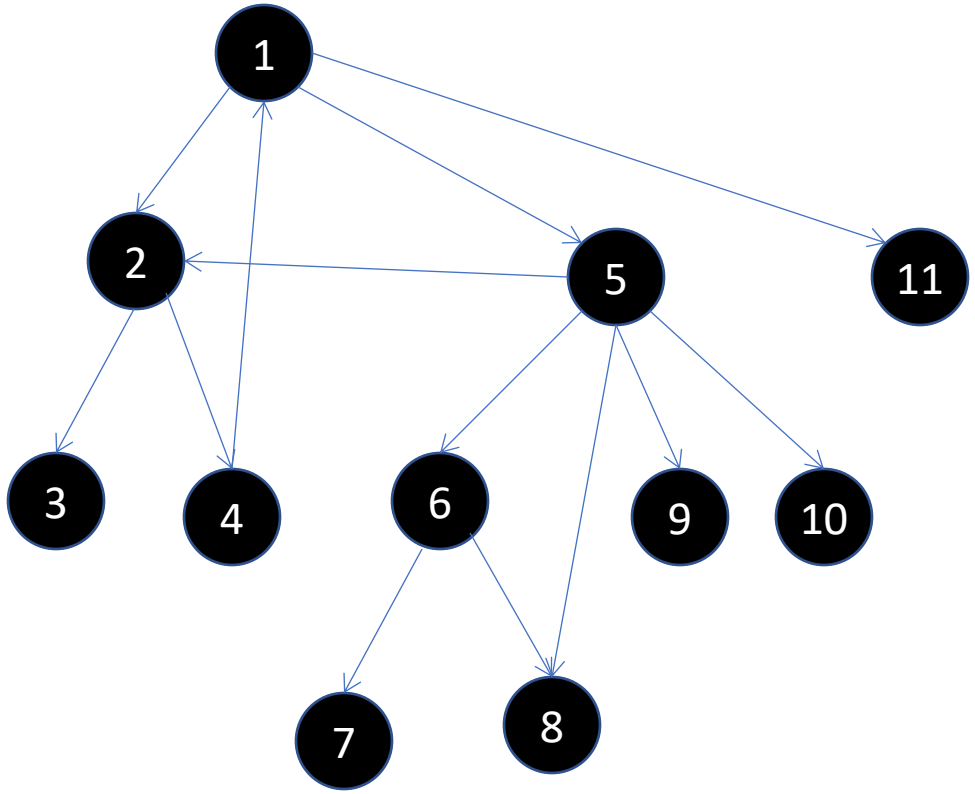


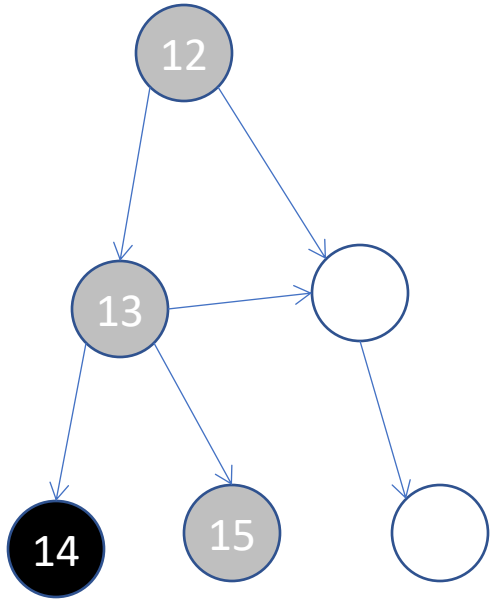



# DFS Traversal

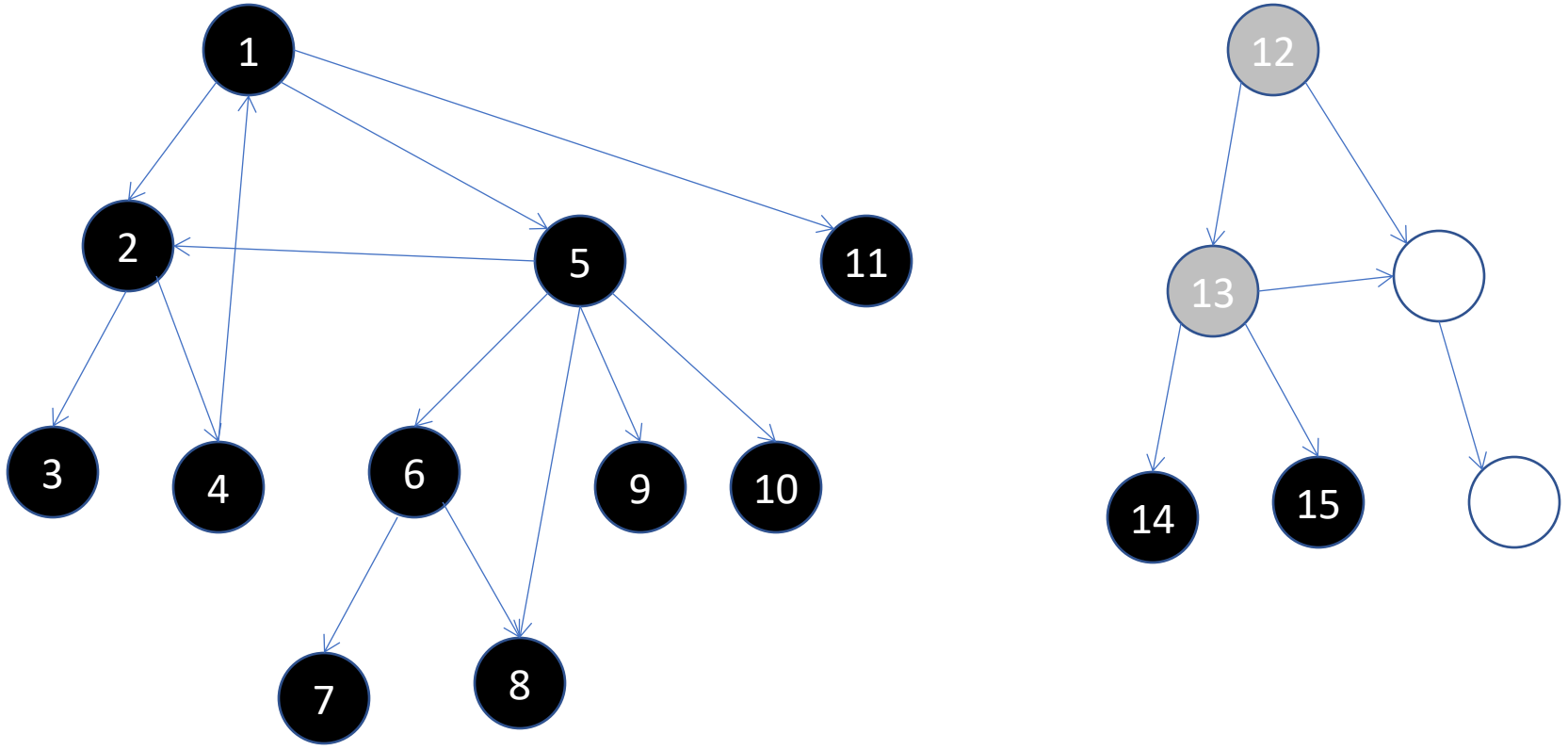


# DFS Traversal

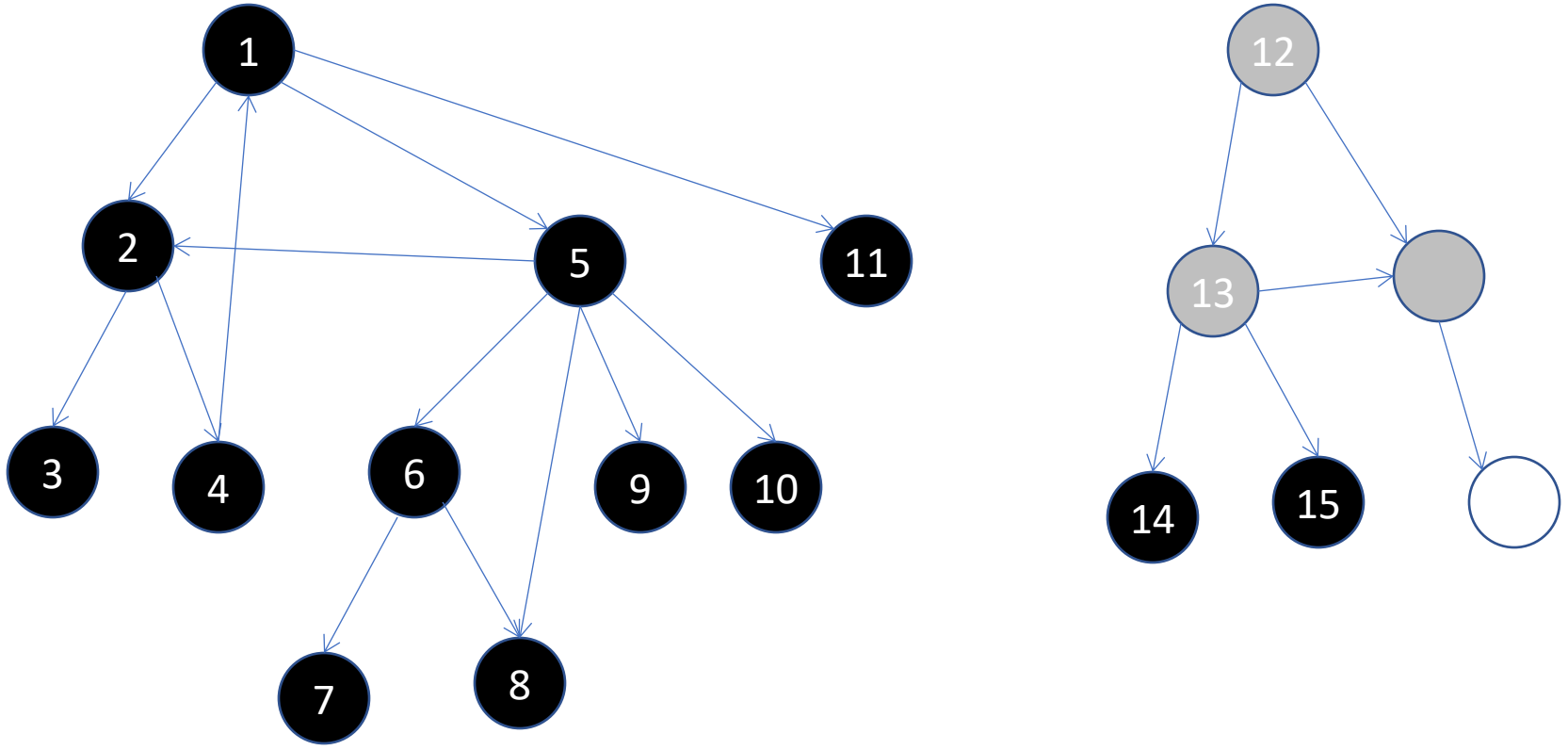


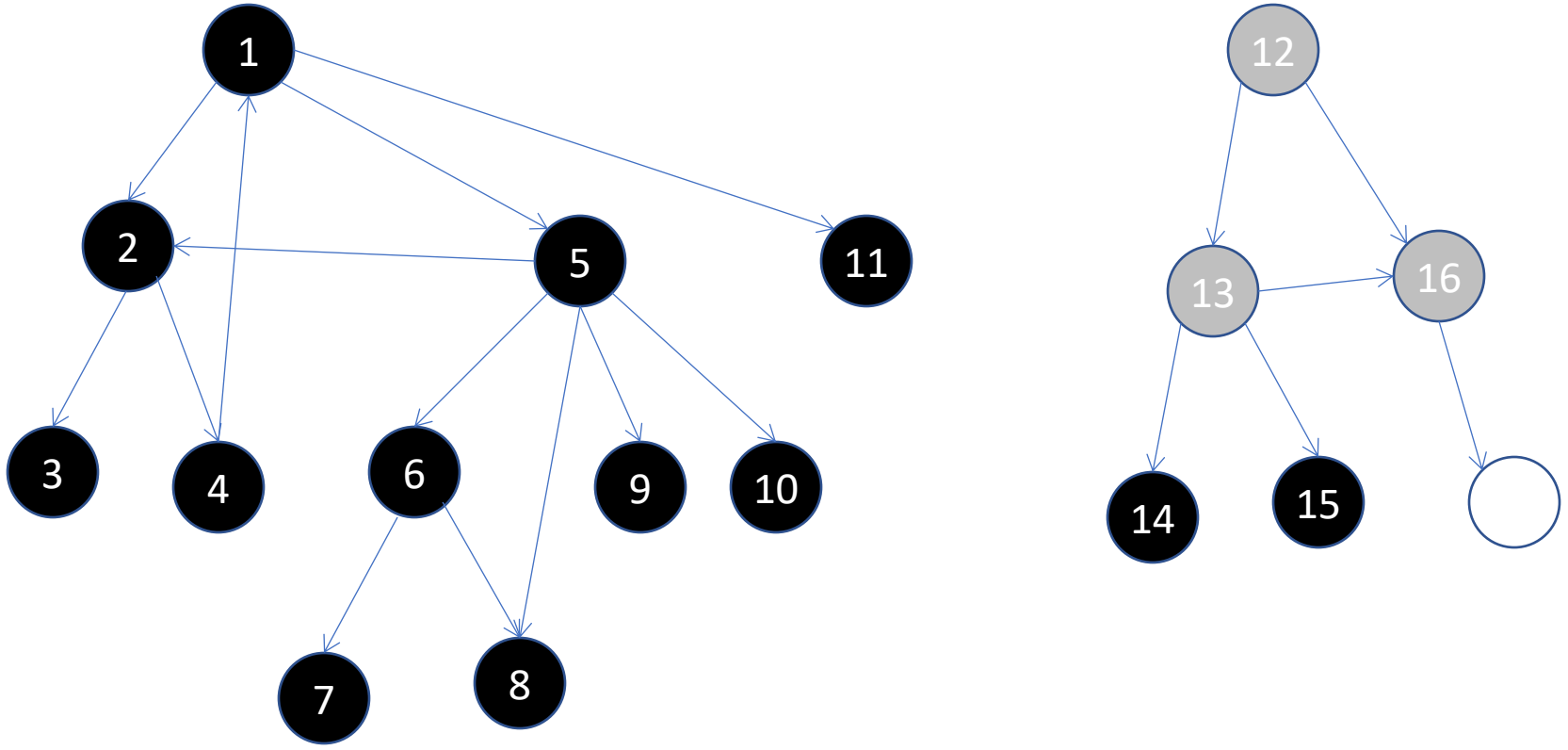
# DFS Traversal



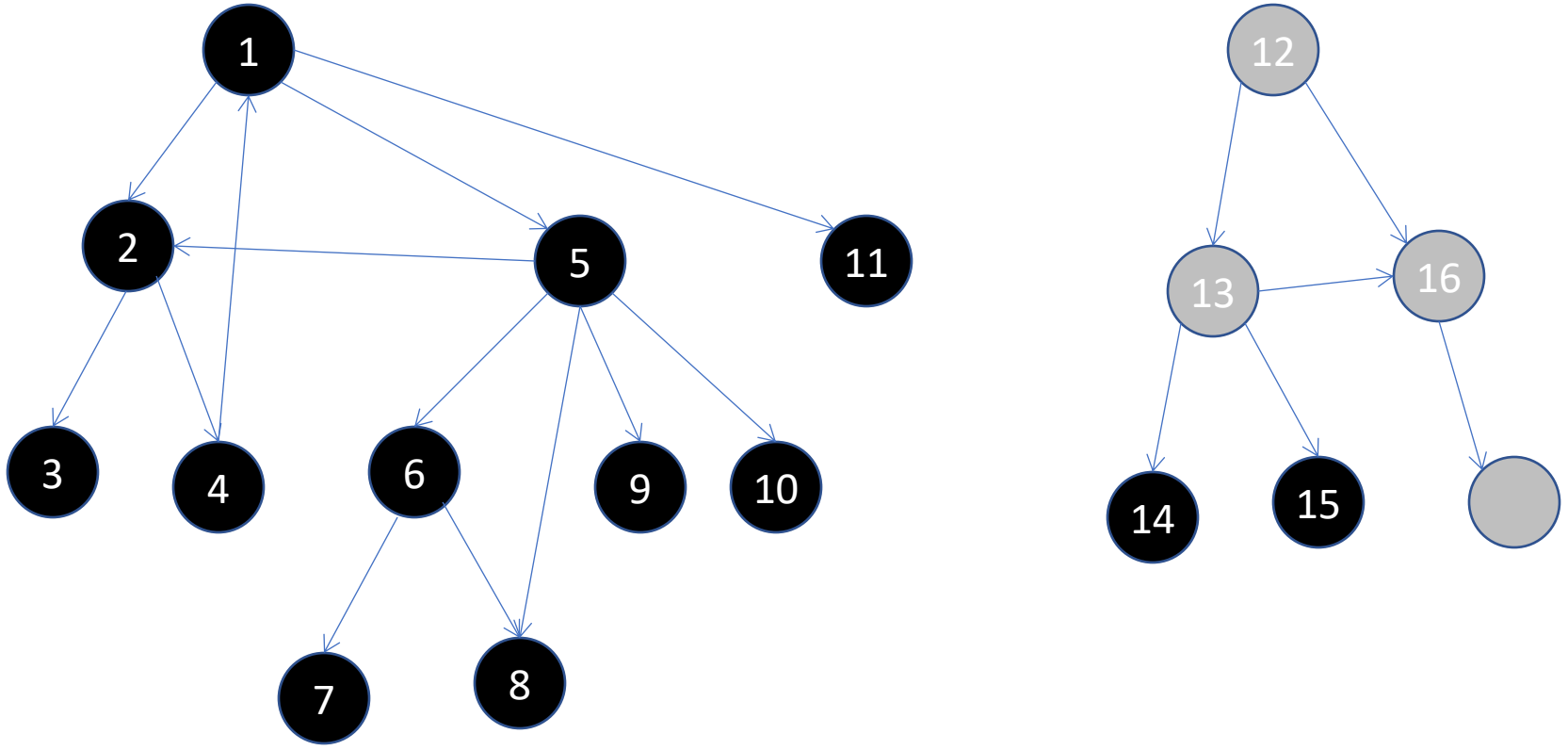
# DFS Traversal



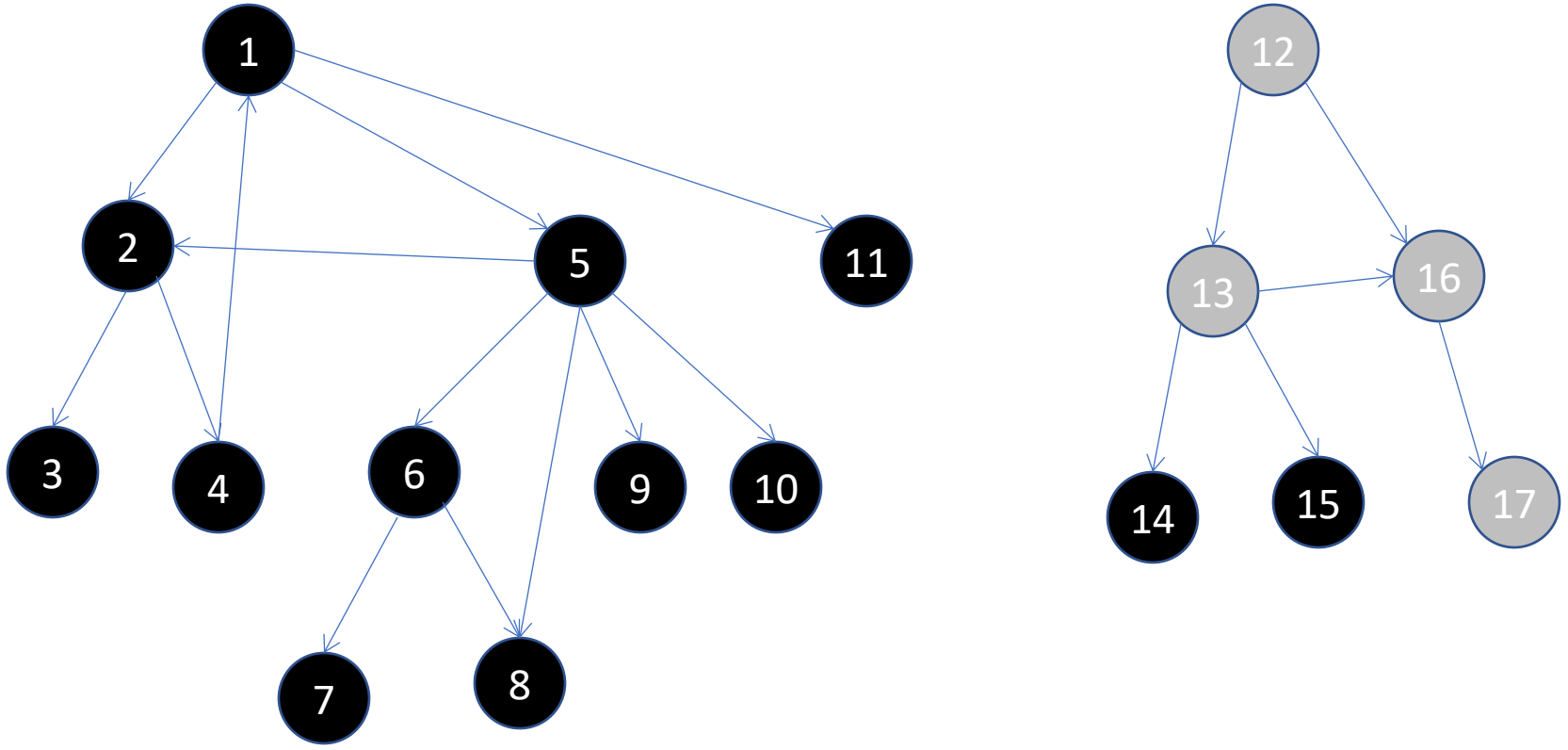
# DFS Traversal



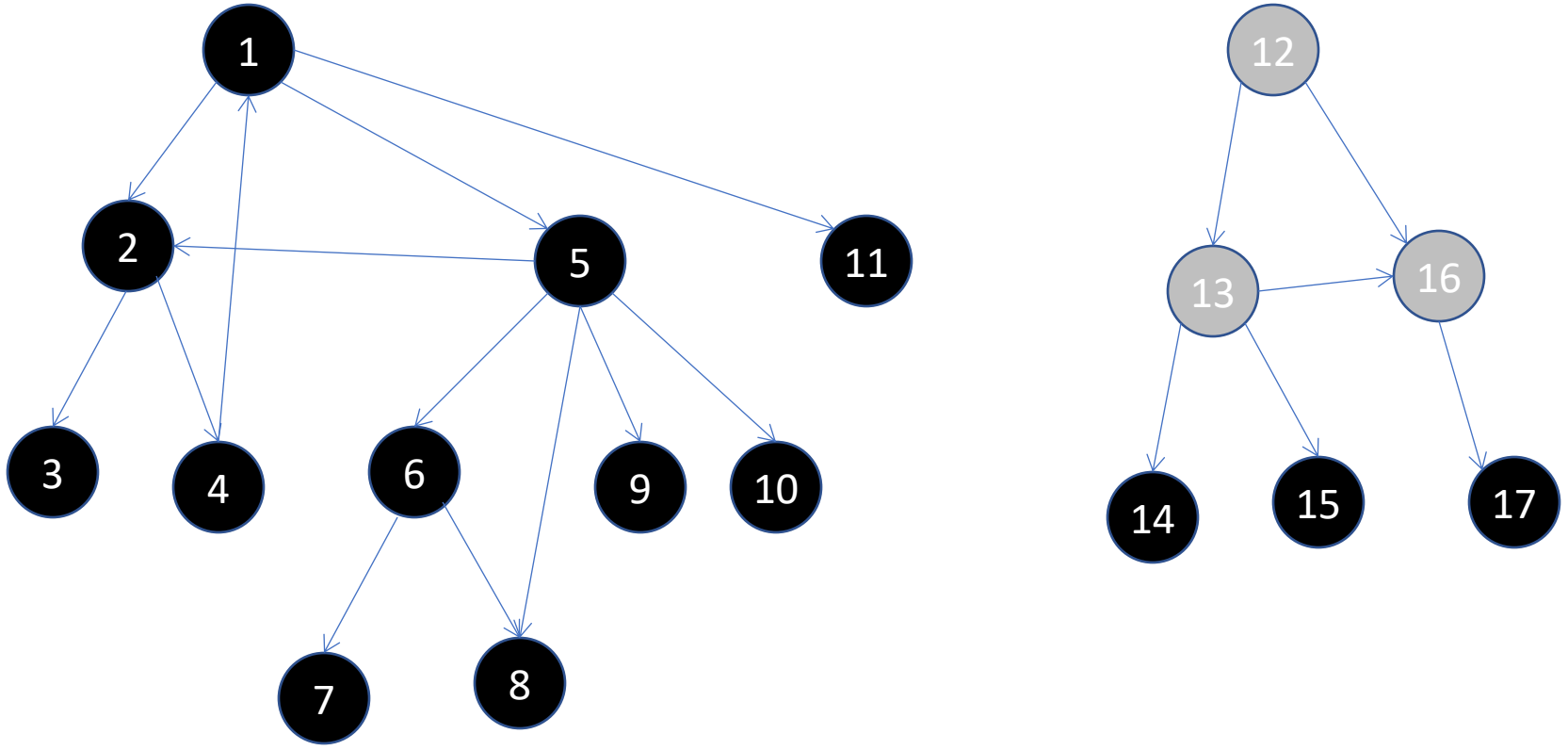
# DFS Traversal



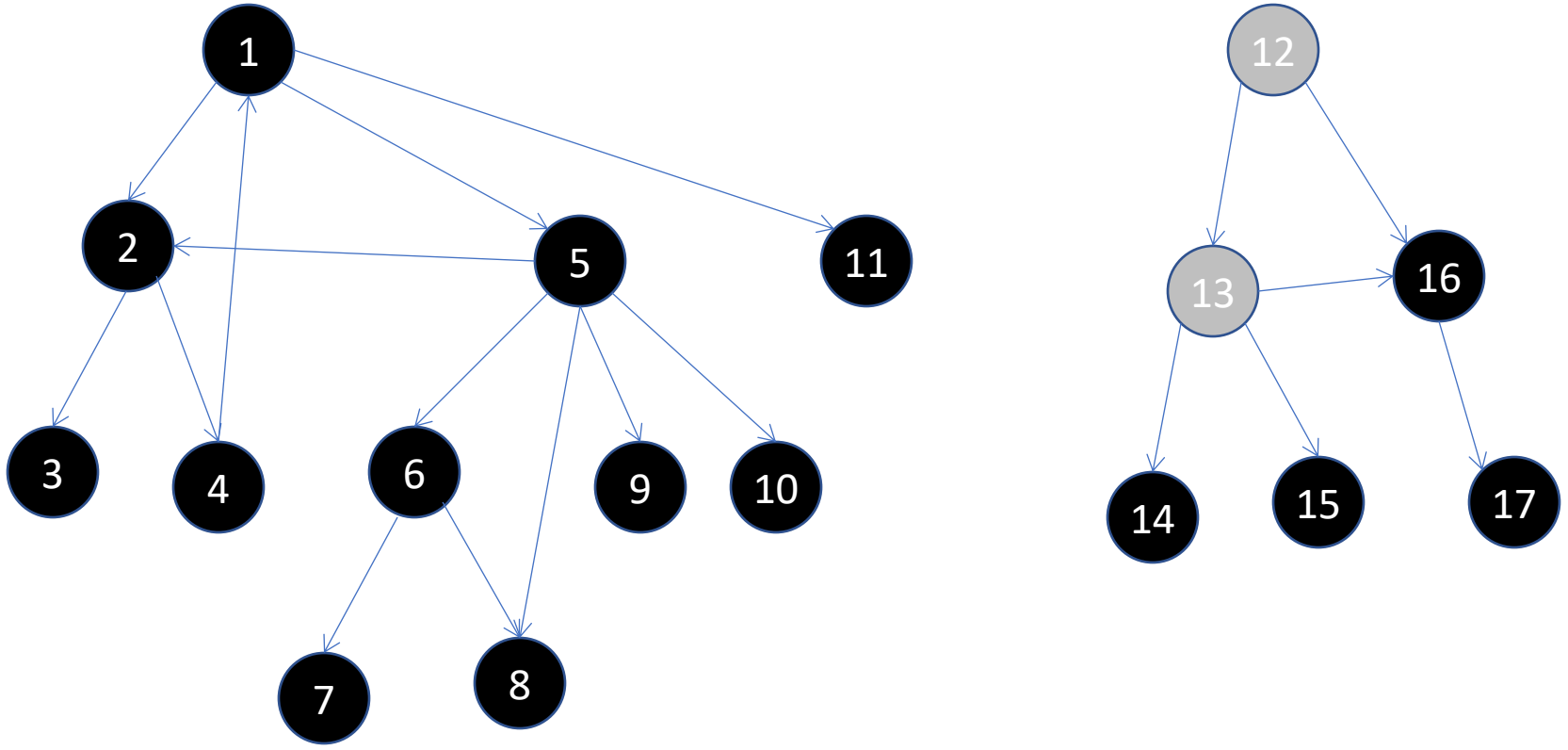
# DFS Traversal



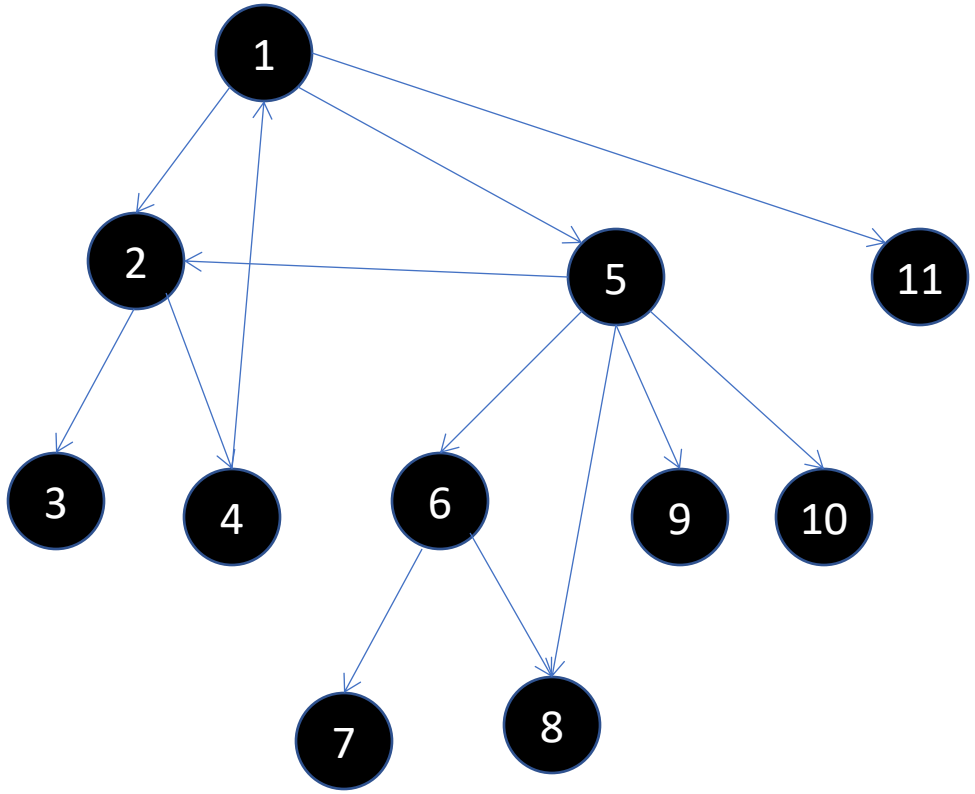
# DFS Traversal

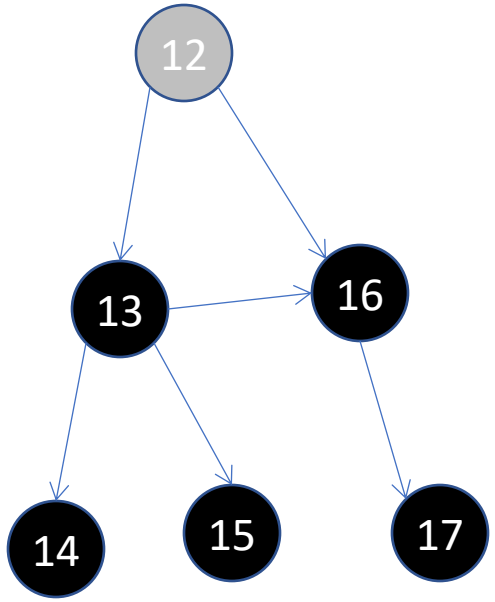


# DFS Traversal

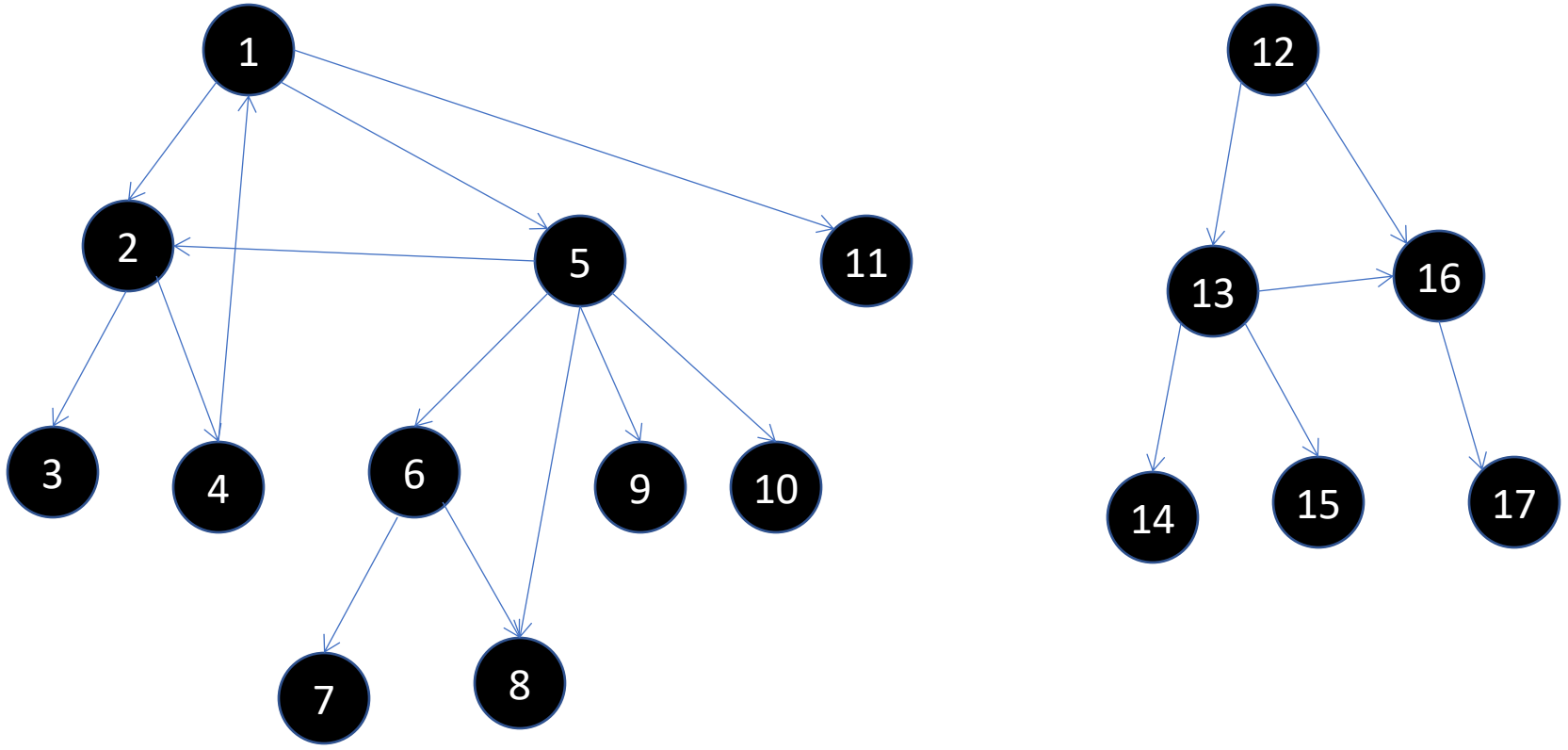


# DFS Traversal



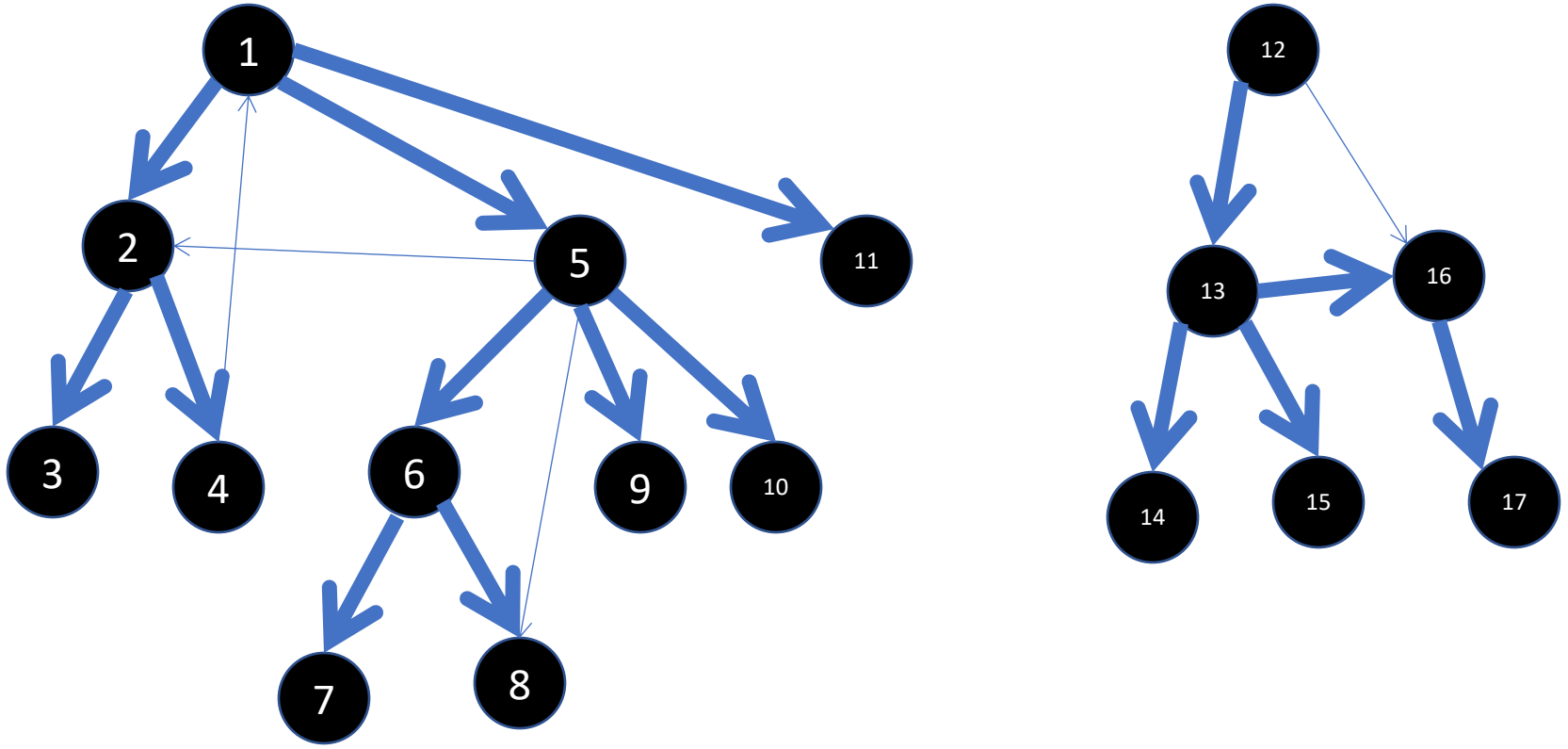
# DFS Traversal



# Four Kinds of Arcs

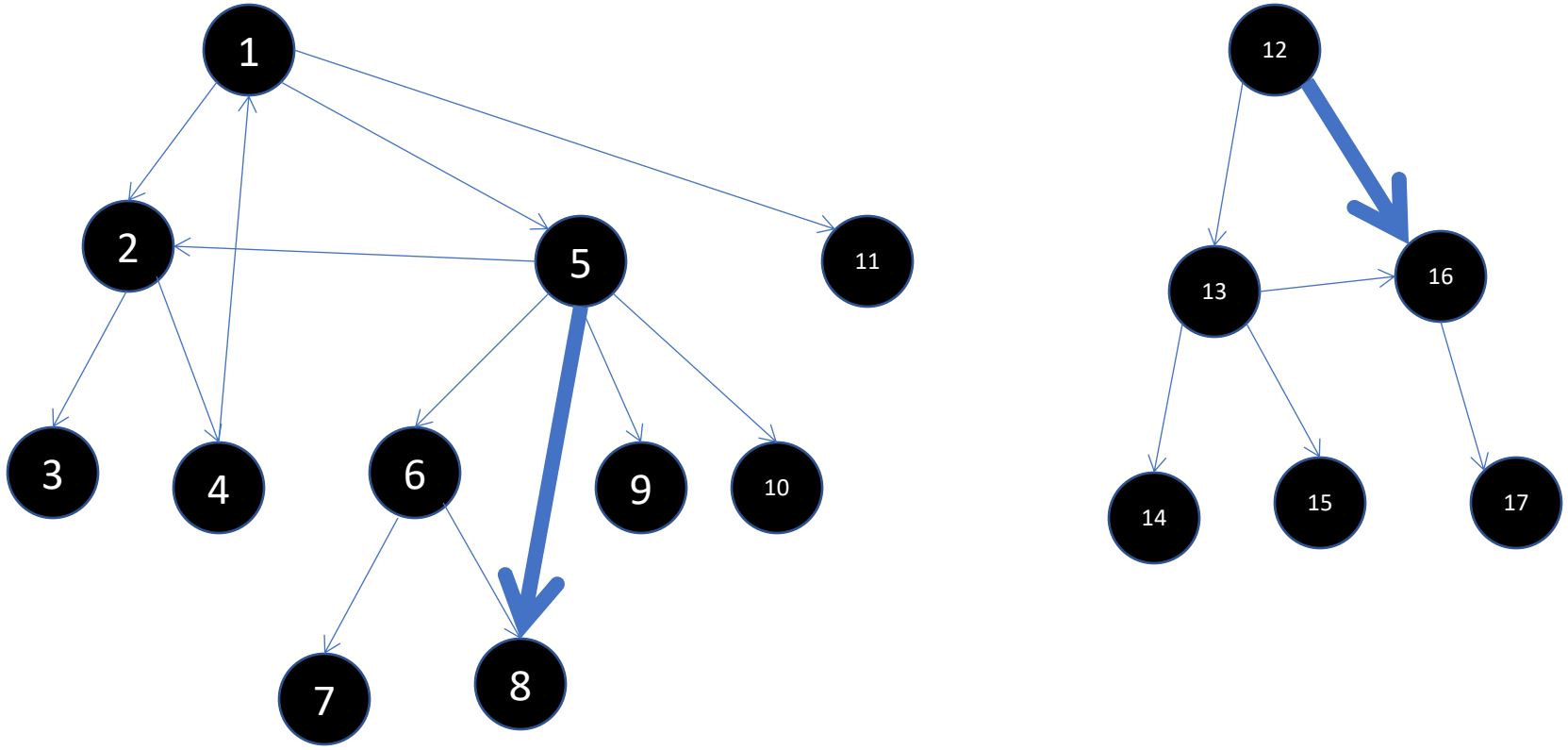
- In a search forest F of a graph G, we can find four different kinds of arcs:
- Tree arc: an arc in G that connects a vertex in G to one of its immediate descendants in the tree of F that the vertex belongs to, i.e., if the arc belongs to the tree
- Forward arc: an arc that does not belong to a tree in F but that connects a vertex to one of its descendants in the tree
- Back arcs: an arc that does not belong to a tree in F but that connects a vertex to one of its ancestors in the tree
- Cross arcs: arcs that fall into neither of the above categories

# DFS Traversal: Tree Arcs



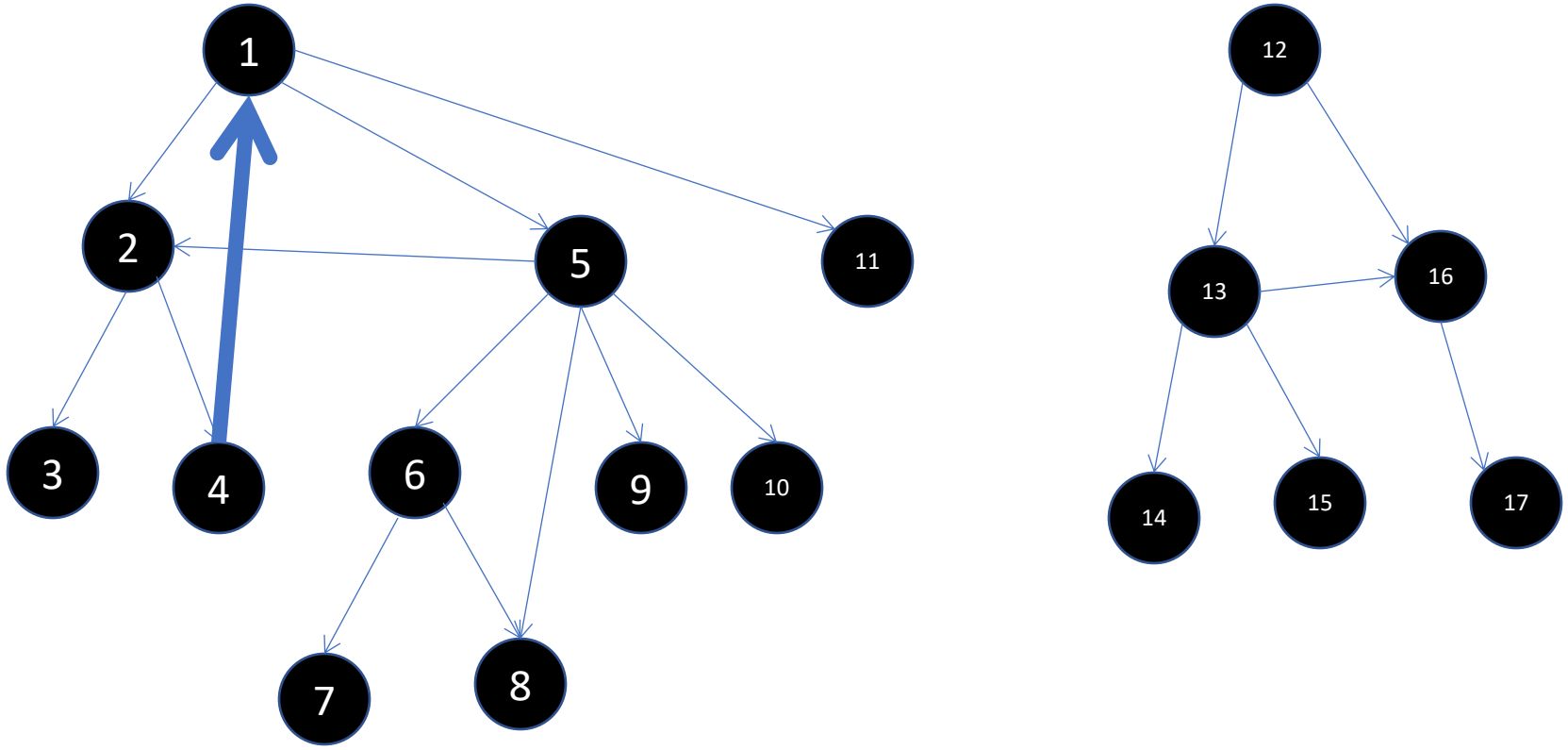
# DFS Traversal: Forward Arcs

**Forward arc**: an arc that does not belong to a tree in $F$ but that connects a vertex to one of its descendants in the tree



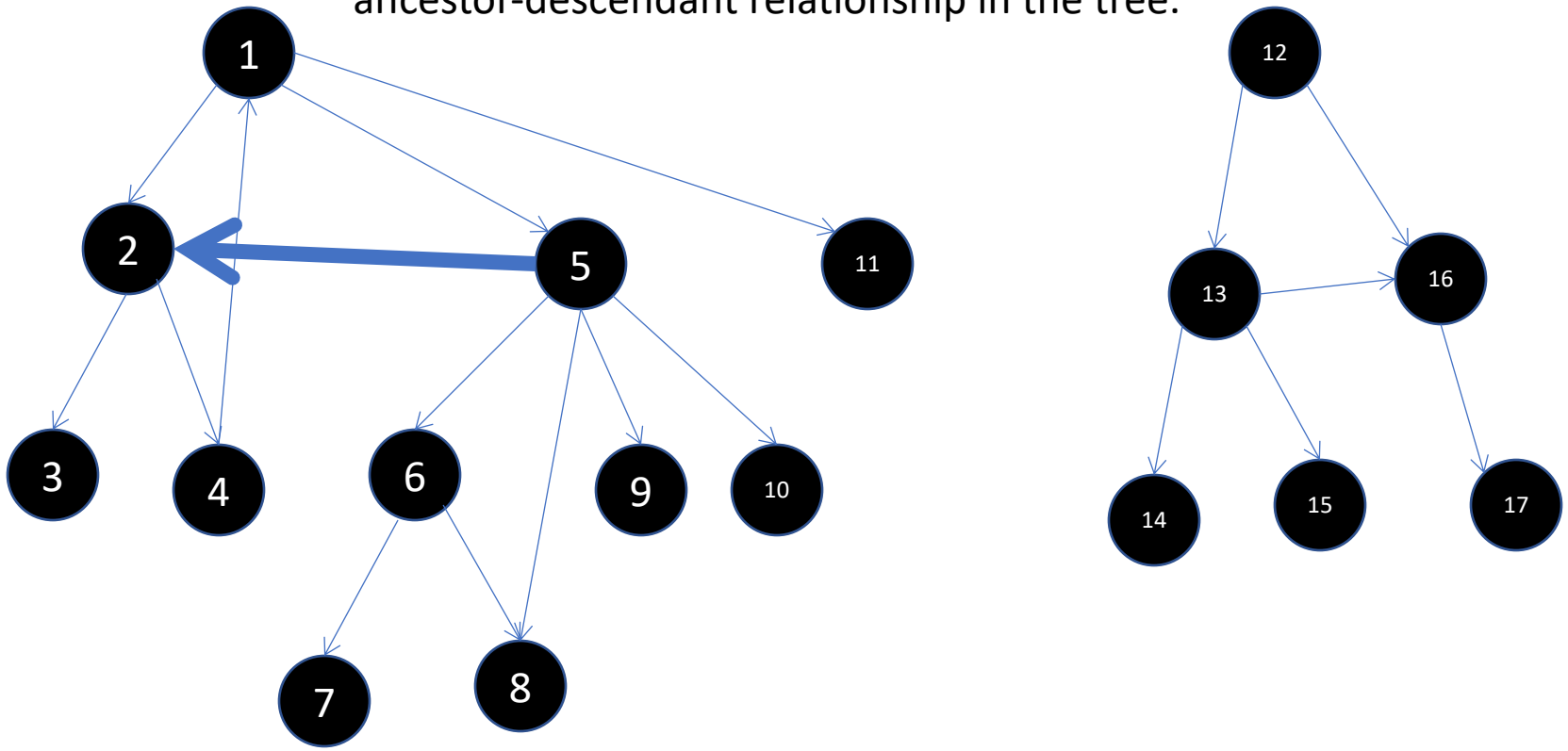
# DFS Traversal: Back Arc

**Back arcs**: an arc that does not belong to a tree in $F$ but that connects a vertex to one of its ancestors in the tree



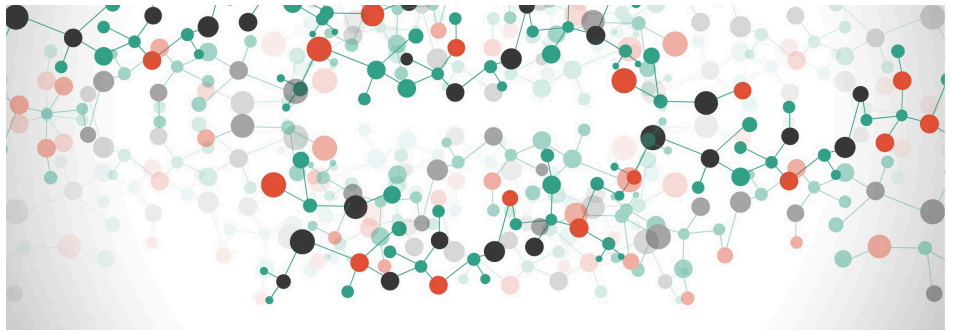
# DFS Traversal: Cross Arc

**Cross arcs**: an arc that does not belong to a tree in $F$ and connects nodes that do not form an ancestor-descendant relationship in the tree.



# OUTLINE

- Graph Traversal Algorithms
  - Depth-first Search (DFS)
  - Breadth-first Search (BFS)
  - Priority-first Search (PFS)
- Implementation
  - Stack – DFS
  - Queue – BFS
  - Priority Queues – PFS

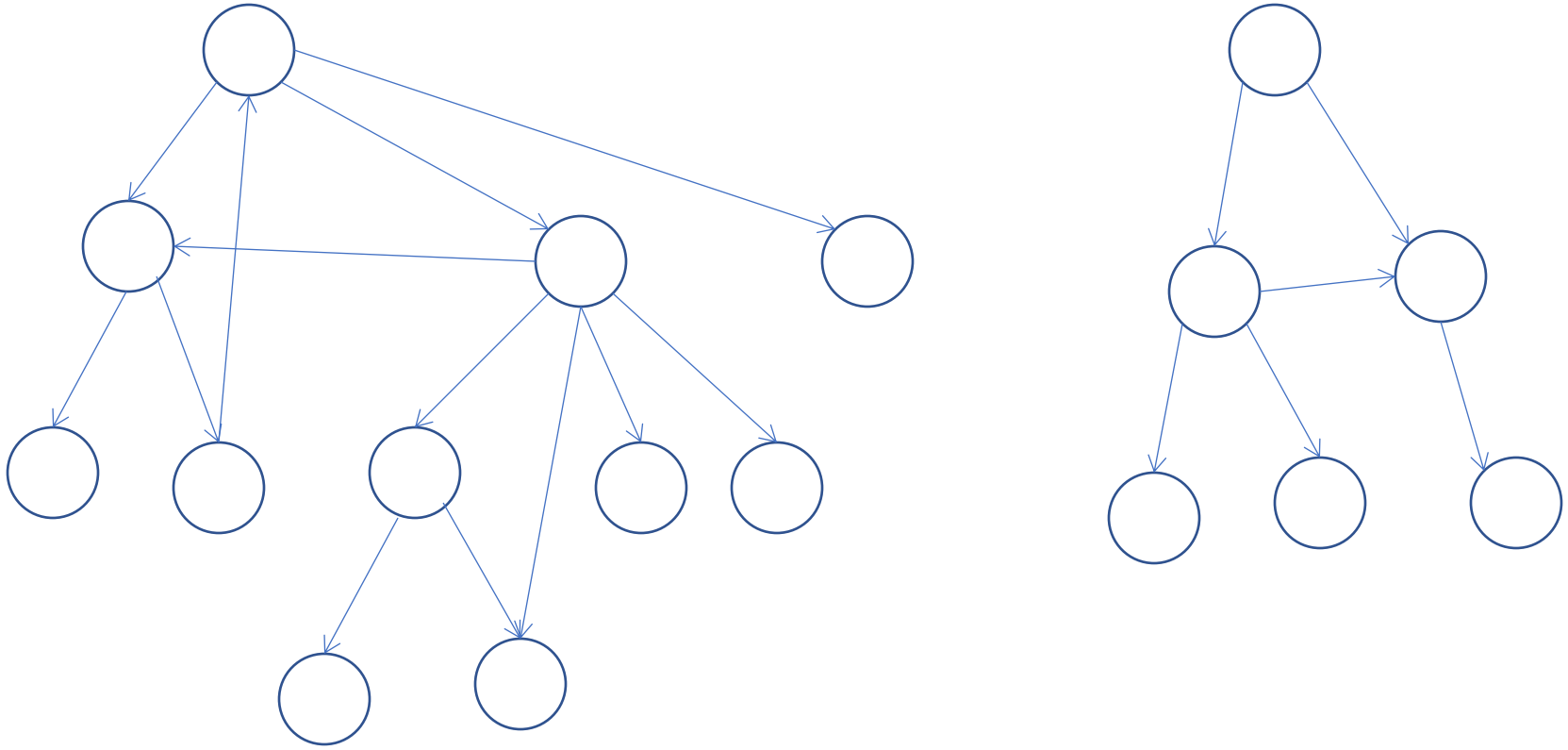


# Breadth-first Search Algorithm (BFS)

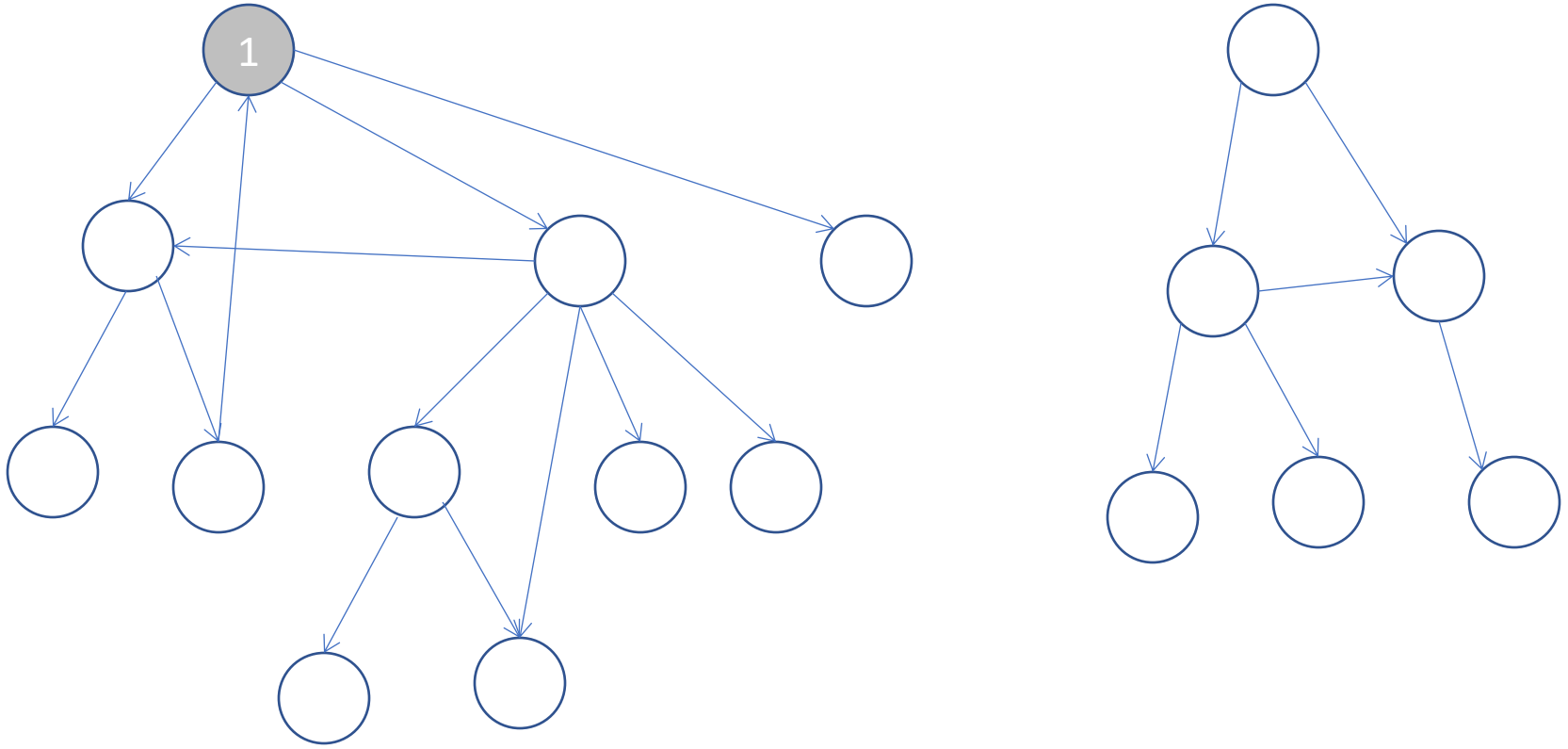
- BFS is also a specific implementation of our fundamental graph traversal algorithm (and also known as breadth-first traversal)

- It specifies that we select the next grey vertex to pick as the oldest remaining grey vertex.

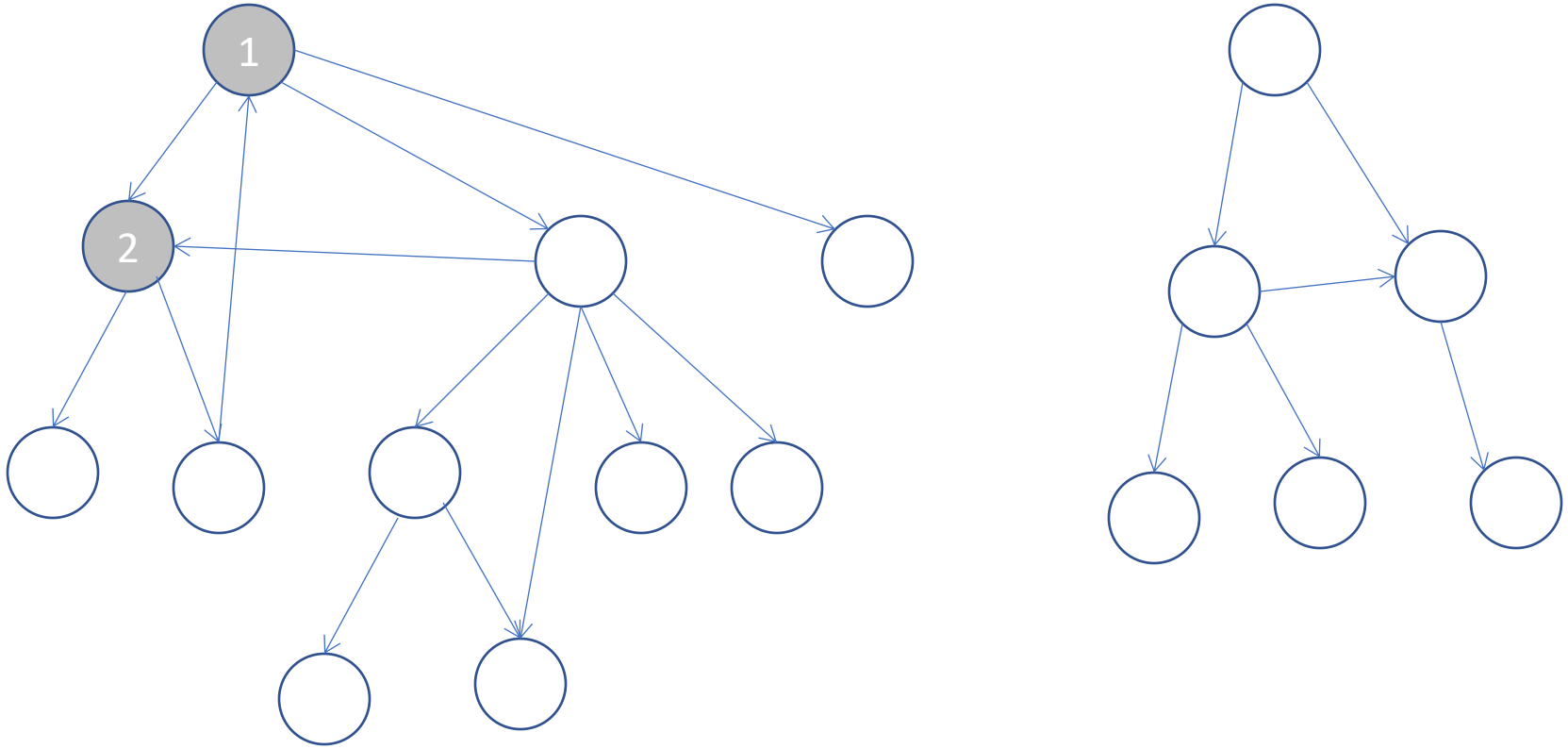
# BFS Traversal



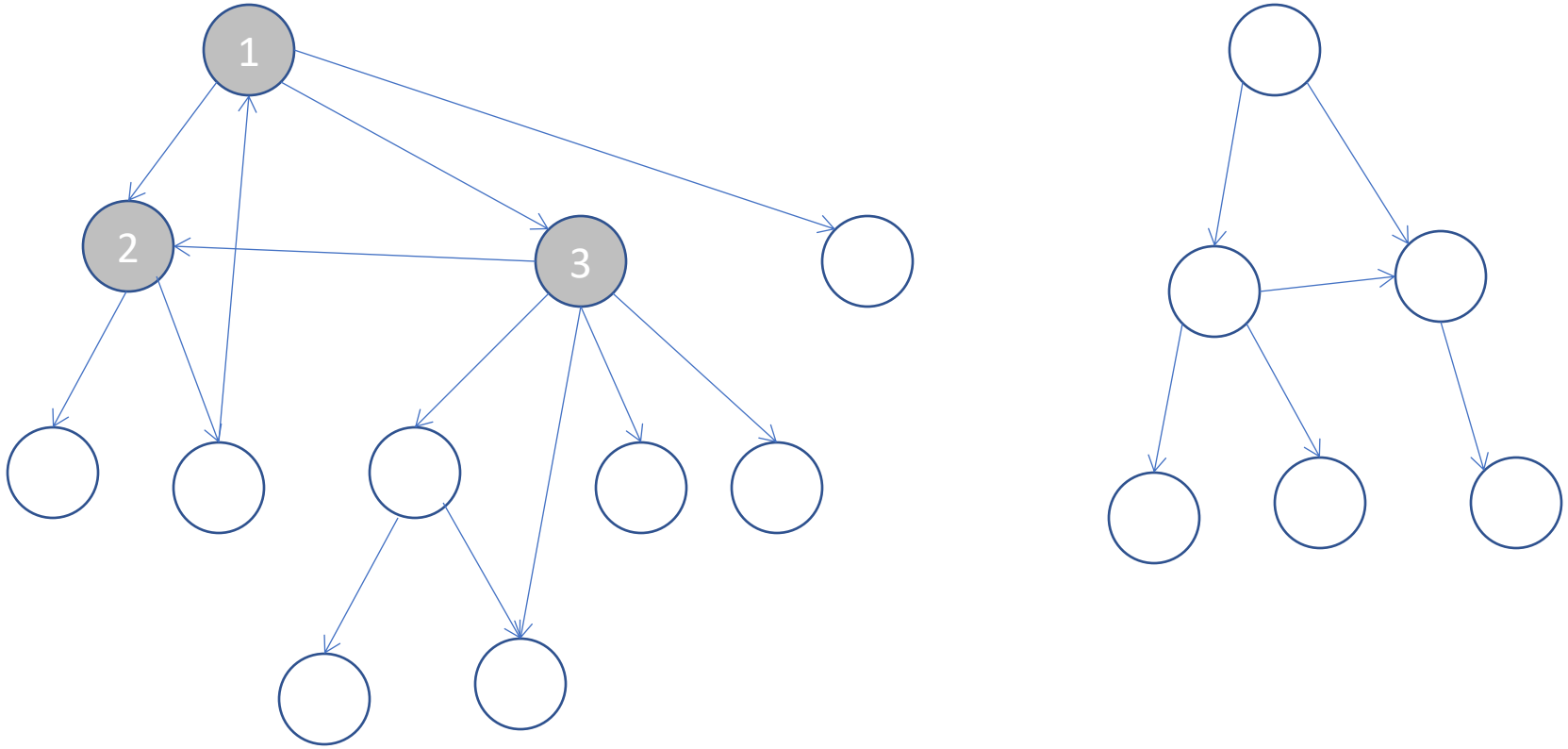
# BFS Traversal



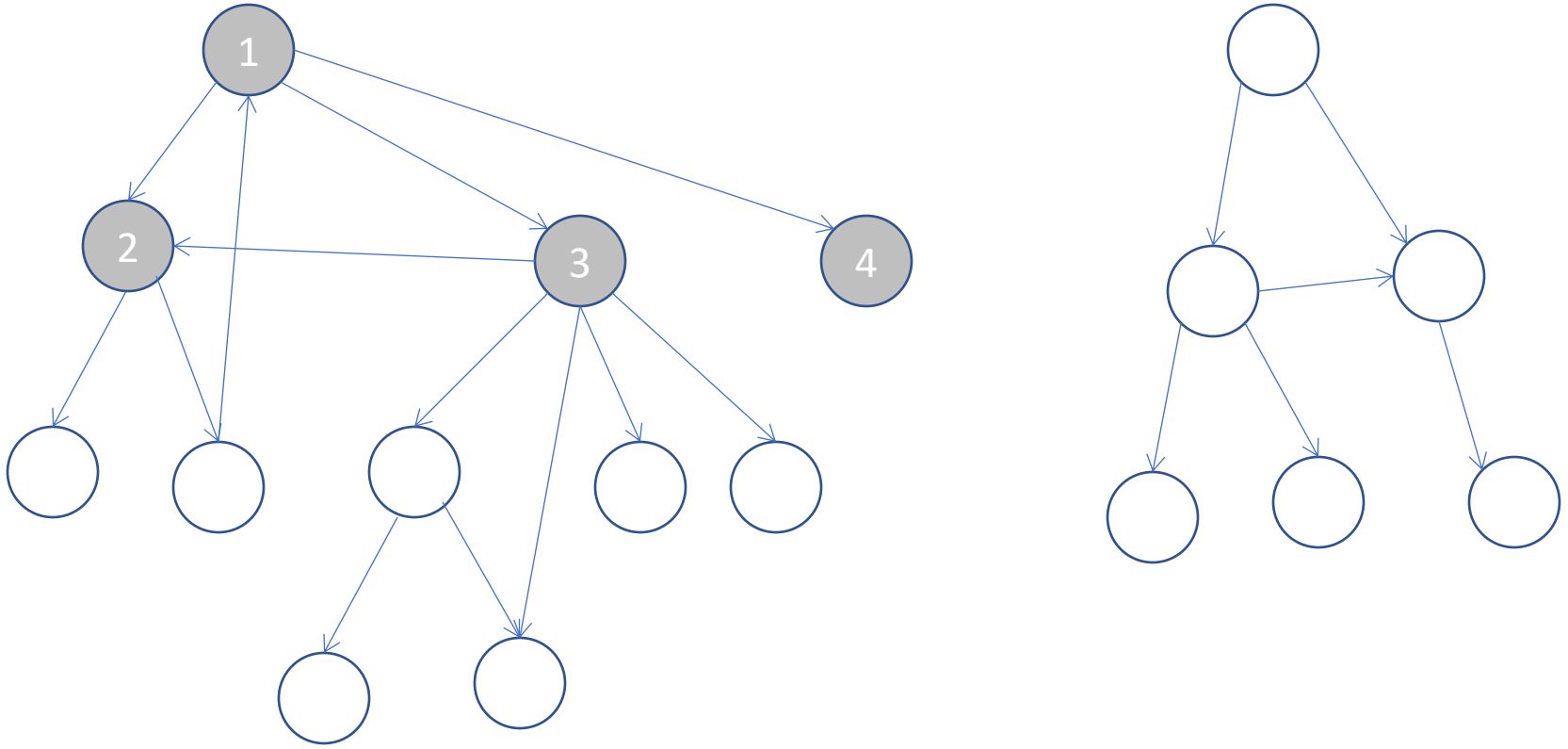
# BFS Traversal



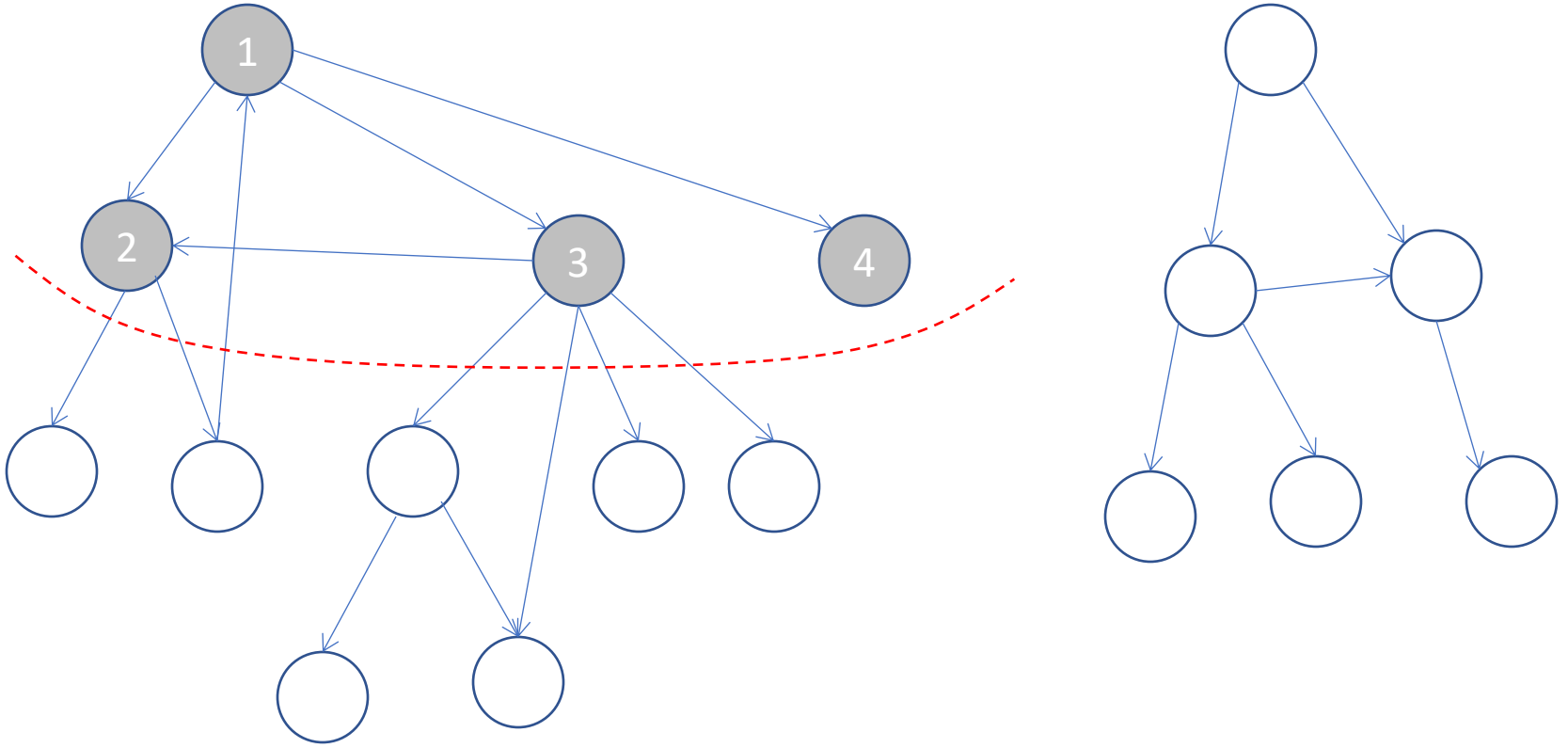
# BFS Traversal



# BFS Traversal

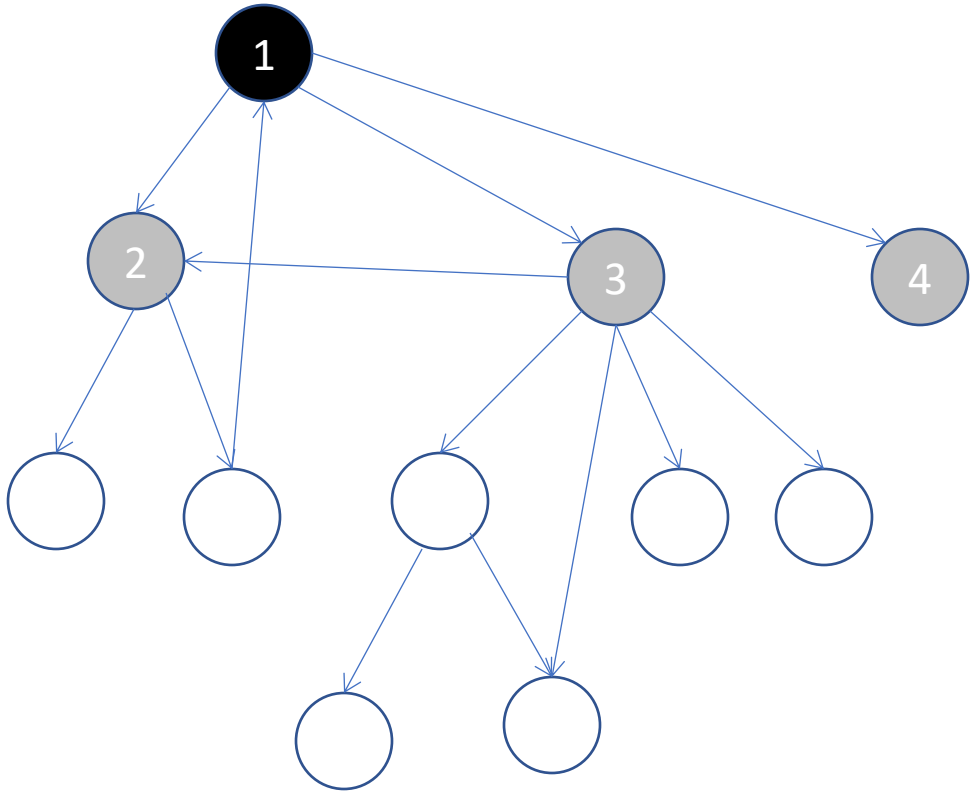


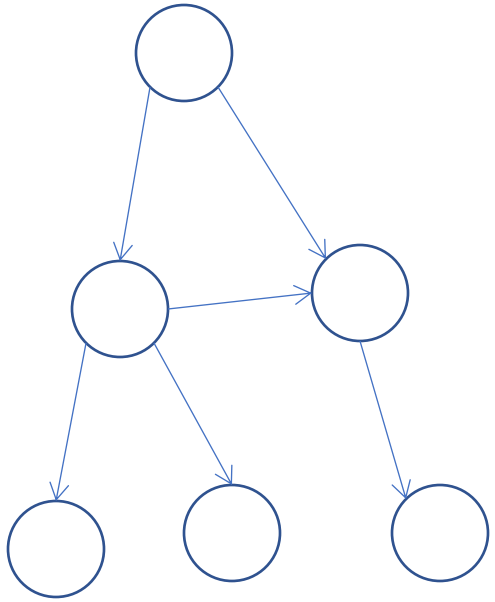
# BFS Traversal



# BFS Traversal

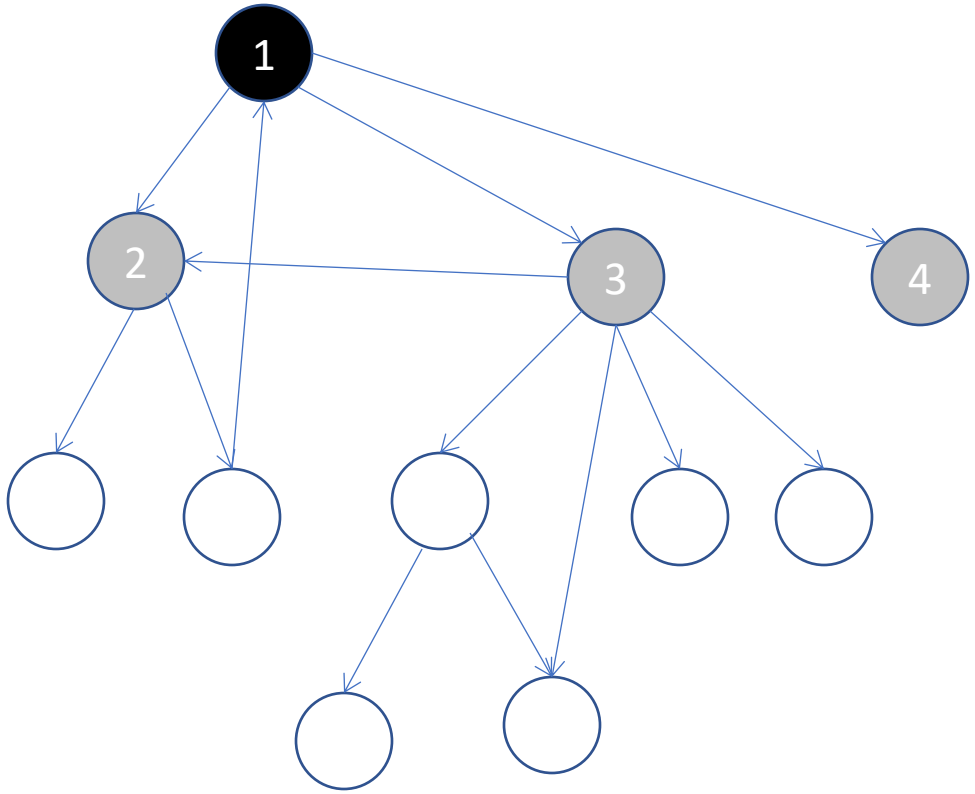
Node 1 does not have any more white neighbour so goes black …

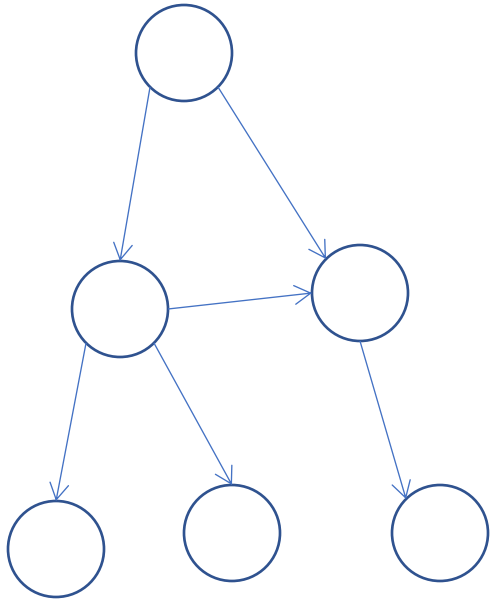


# BFS Traversal

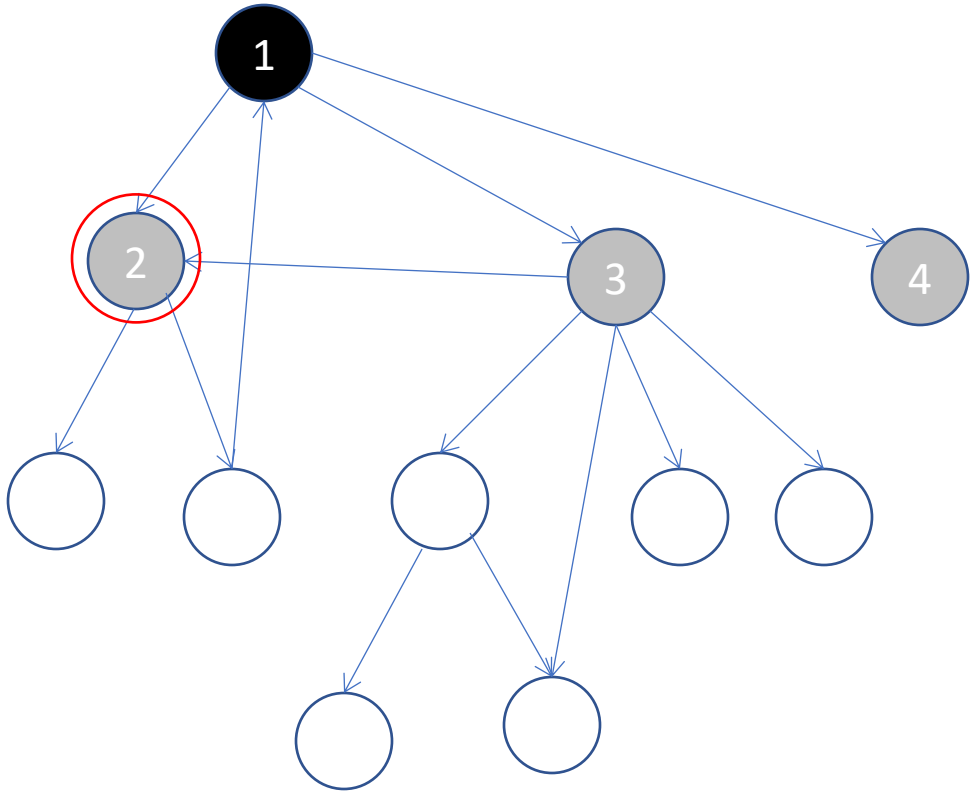
Next grey node the inner loop selects is the oldest one. Which is it?

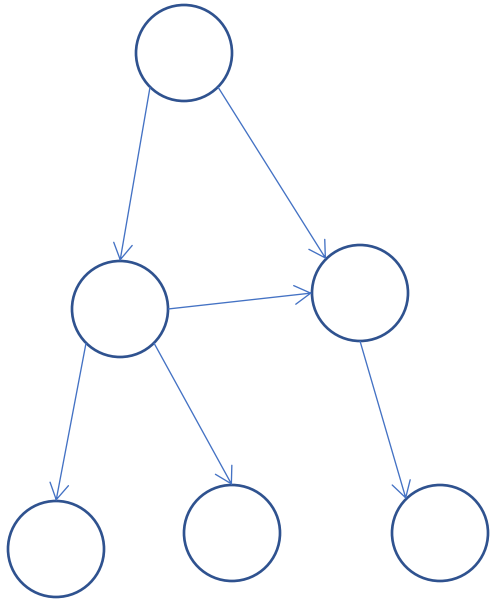


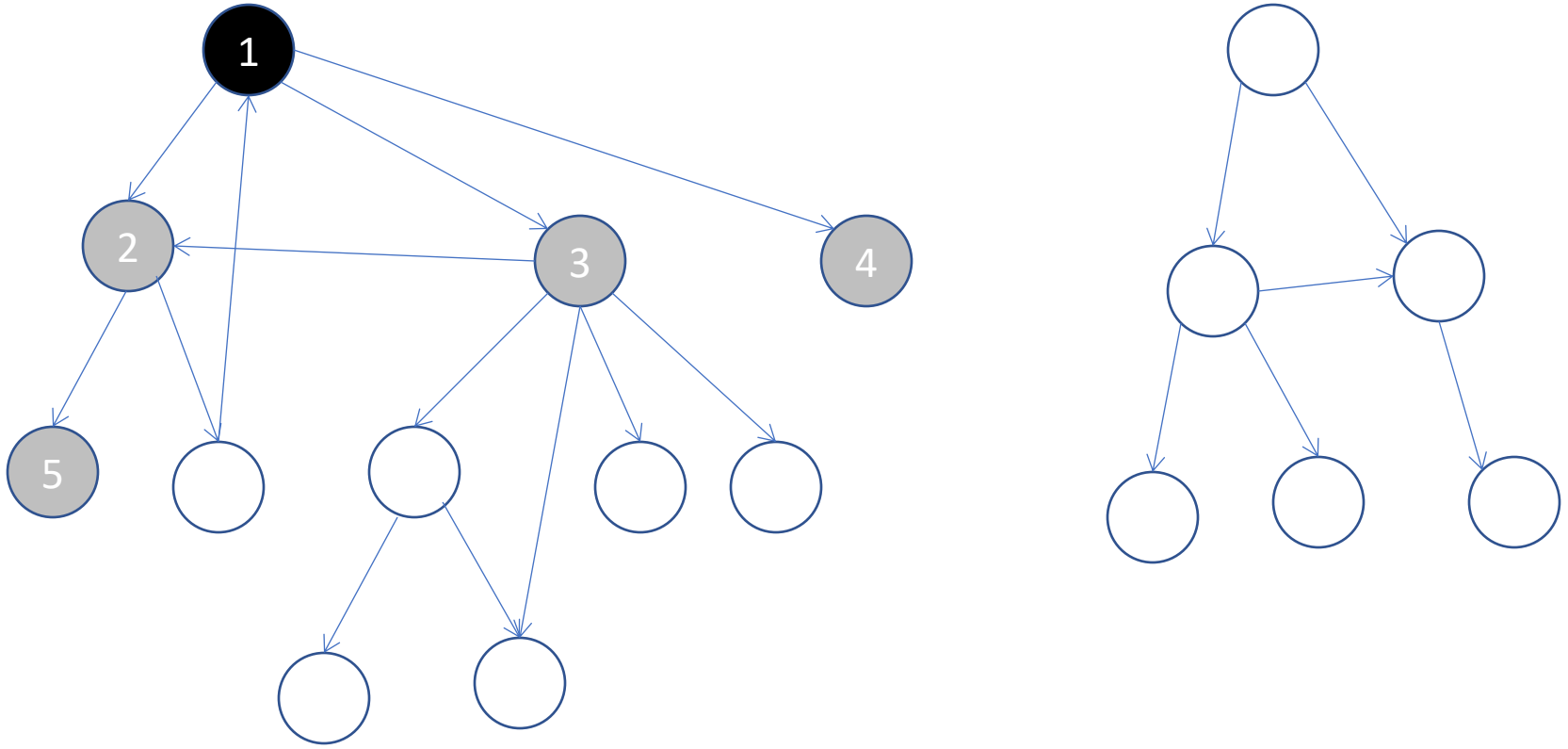
# BFS Traversal

Next grey node the inner loop selects is the oldest one. 2

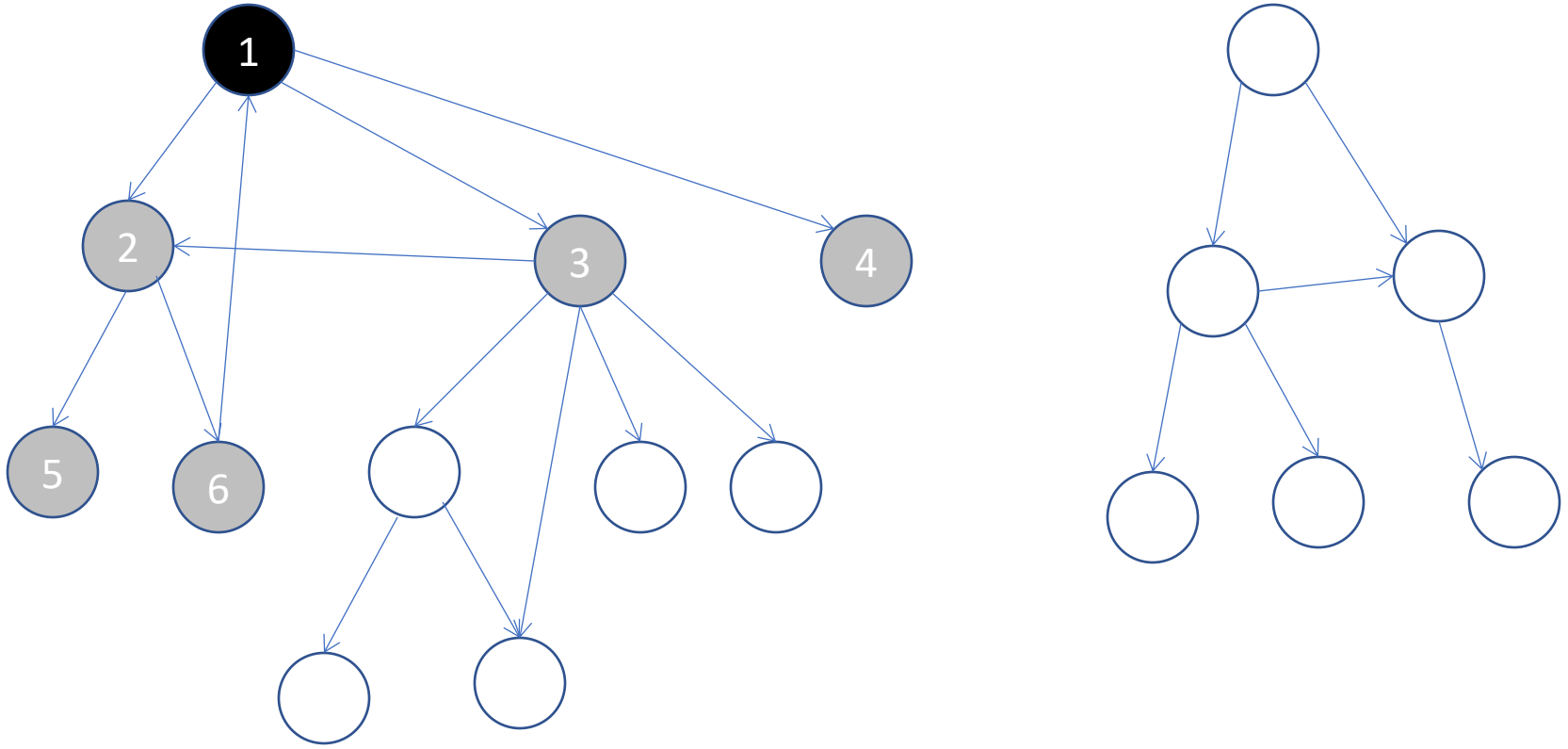


# BFS Traversal

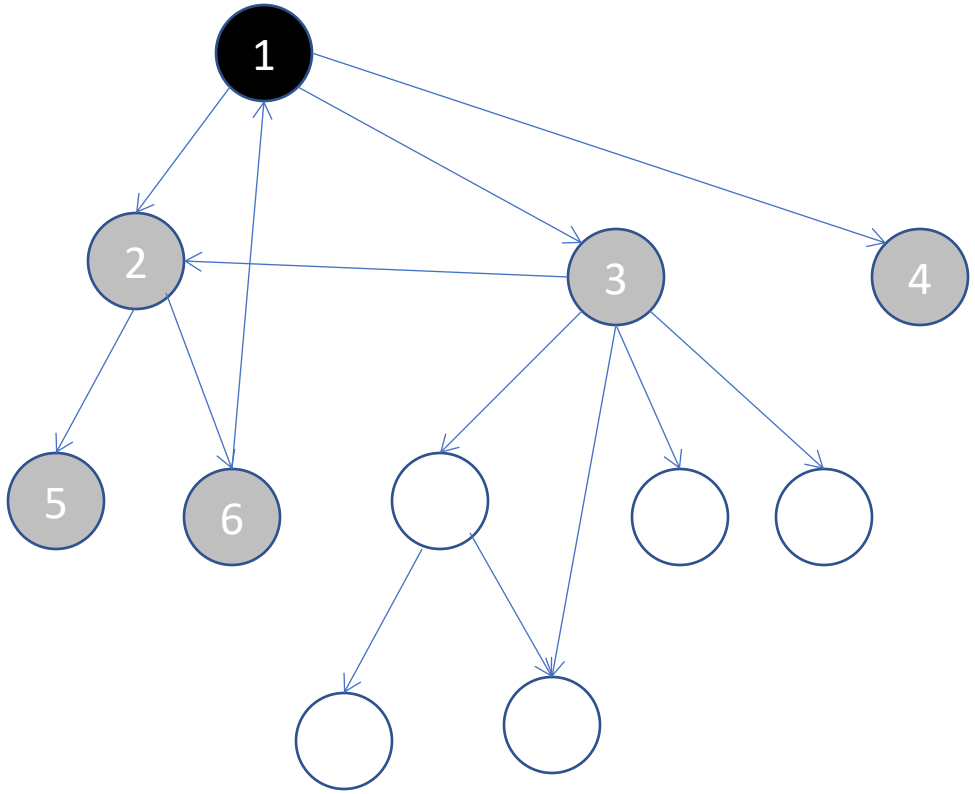


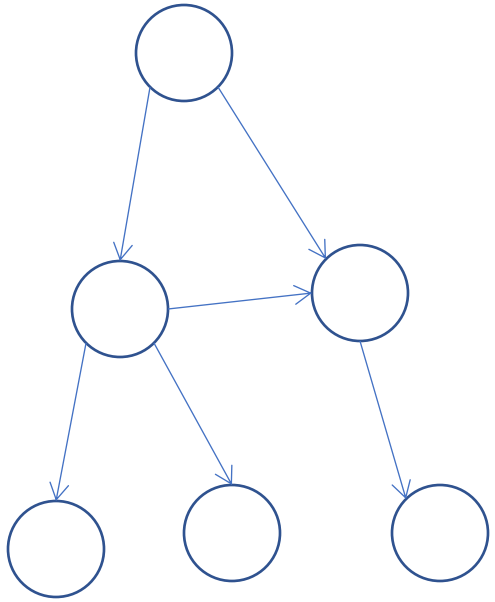
# BFS Traversal



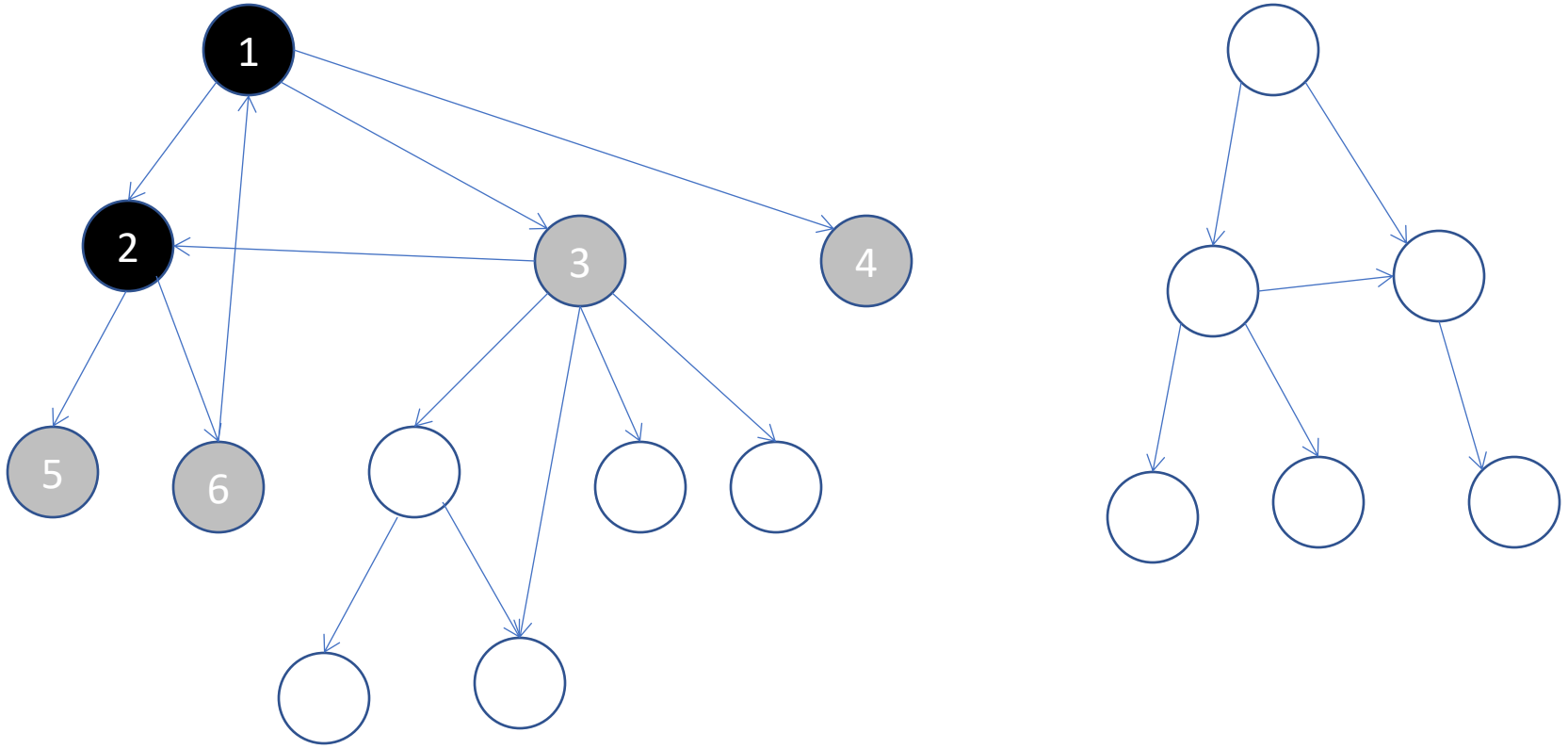
# BFS Traversal

Any more white nodes from 2?

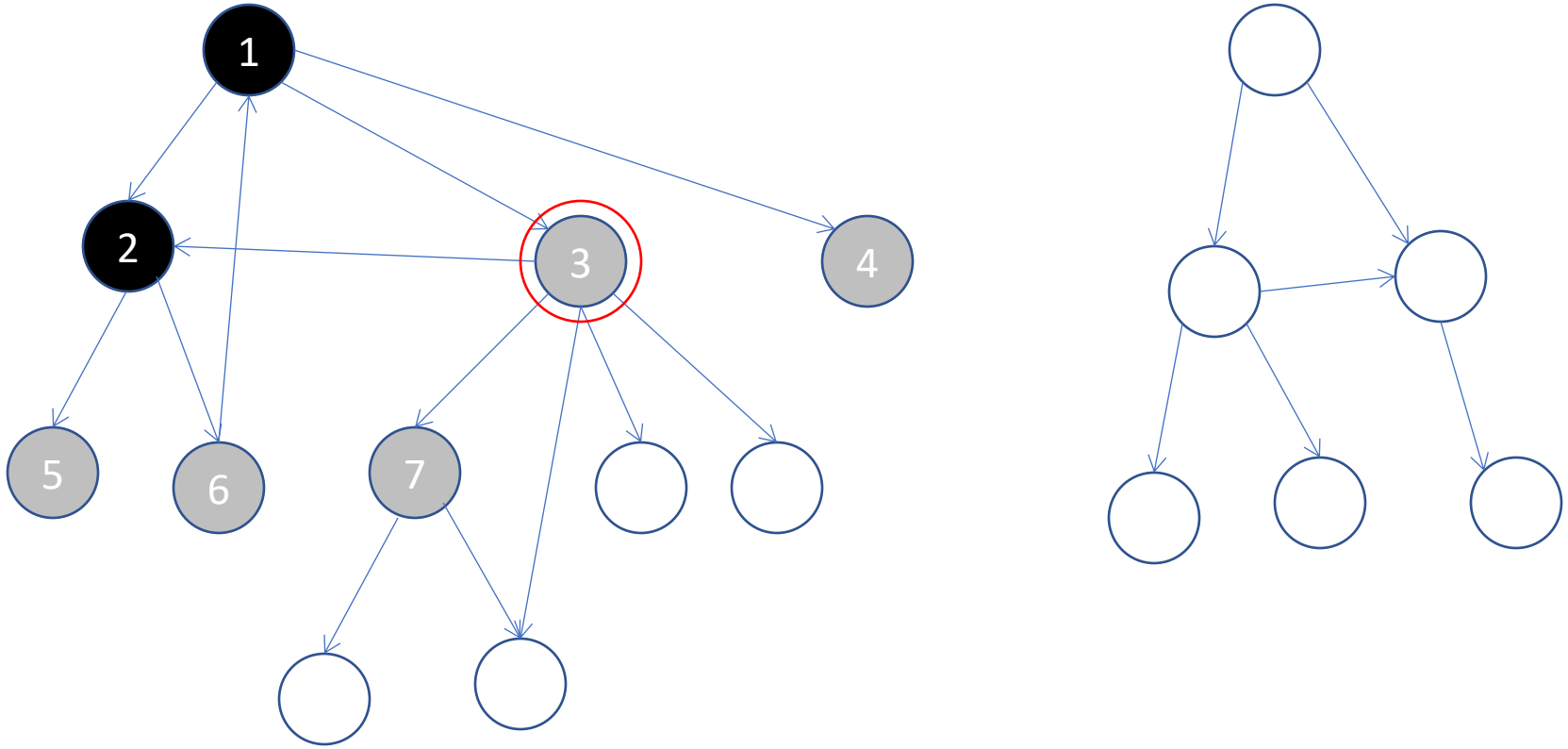


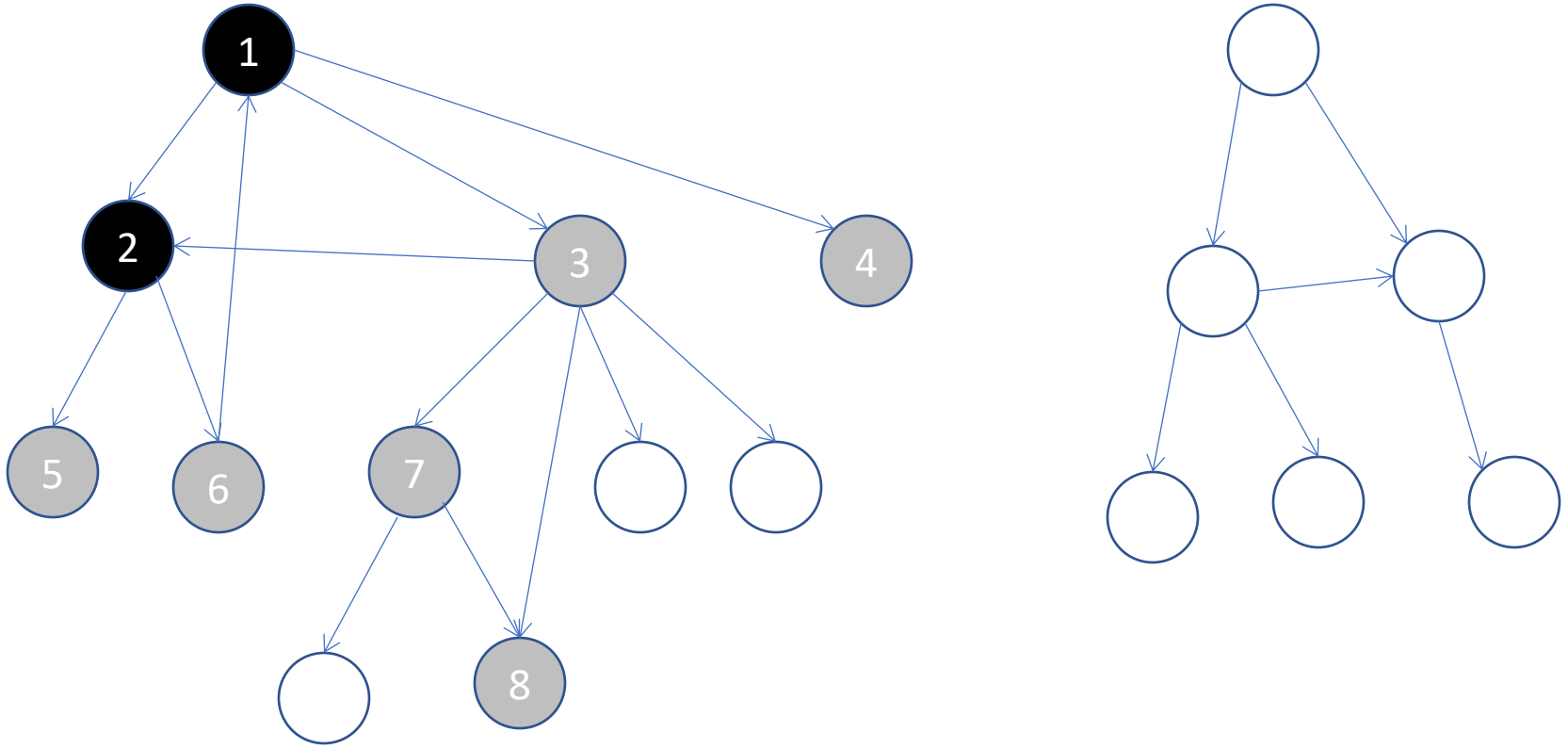
# BFS Traversal



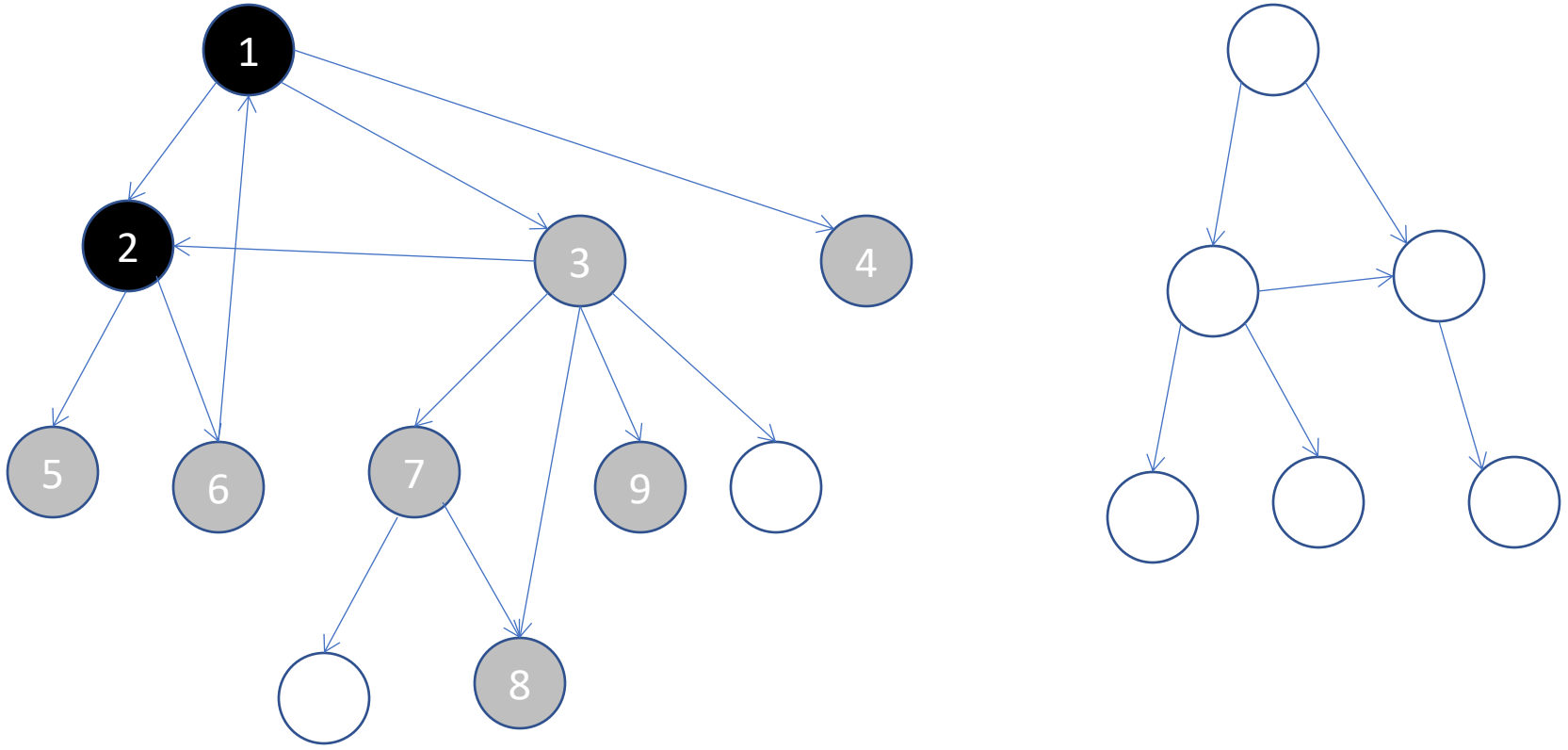
# BFS Traversal



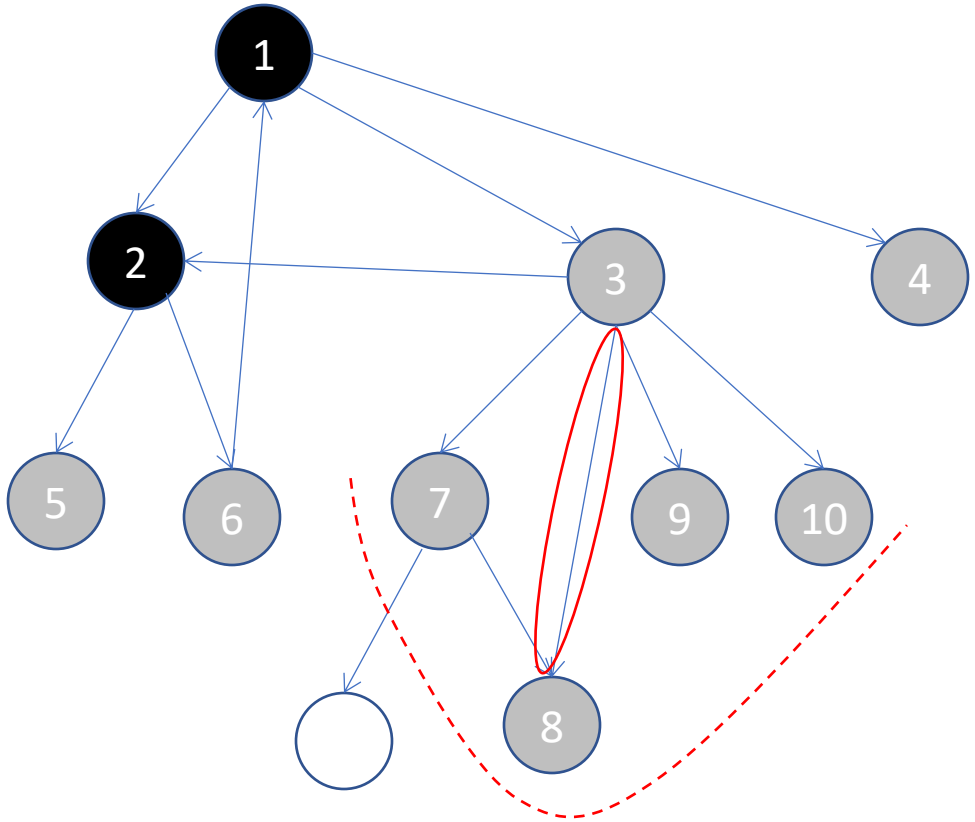
# BFS Traversal

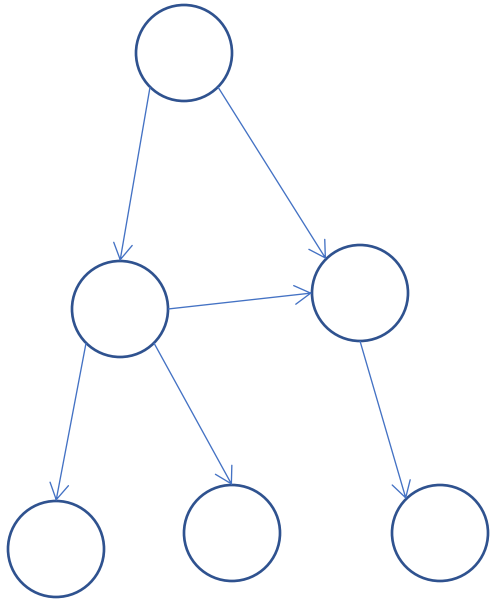


# BFS Traversal

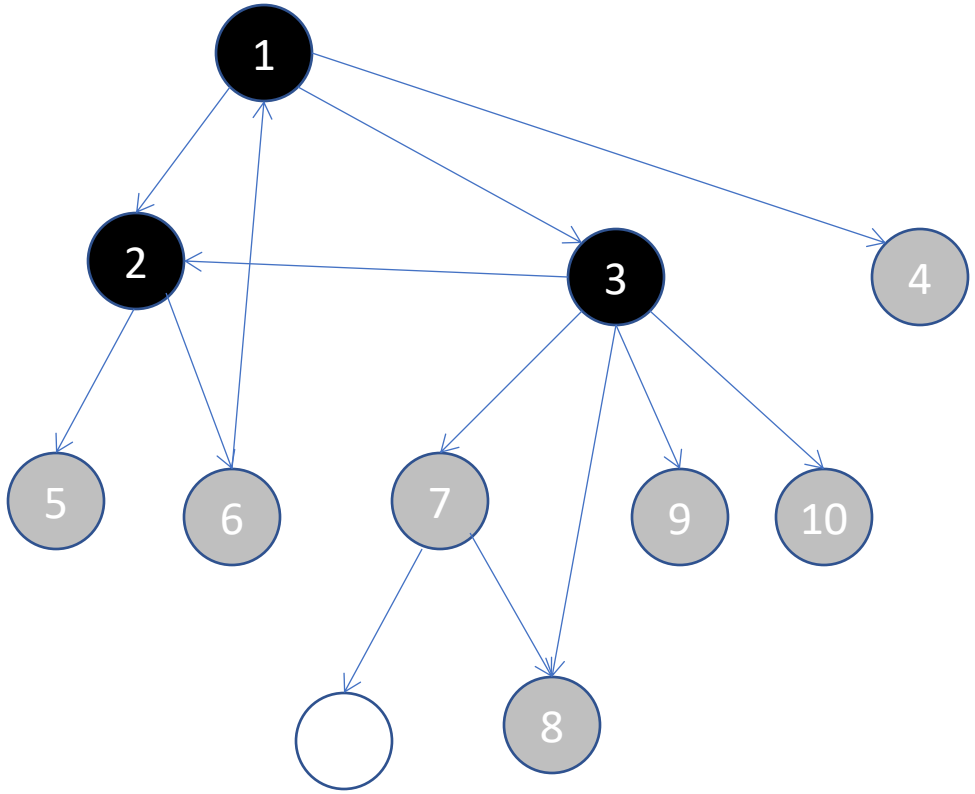


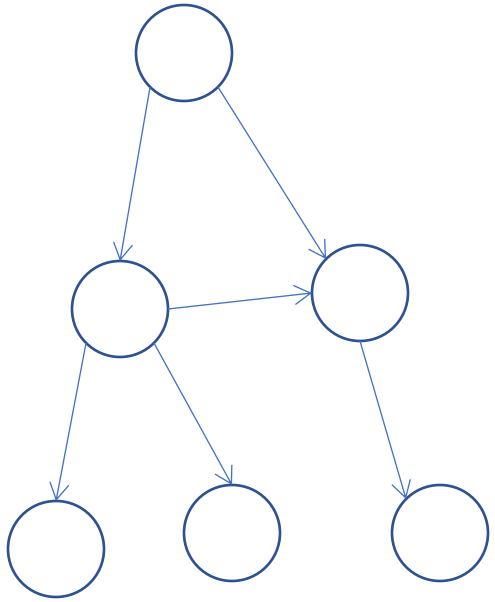
# BFS Traversal



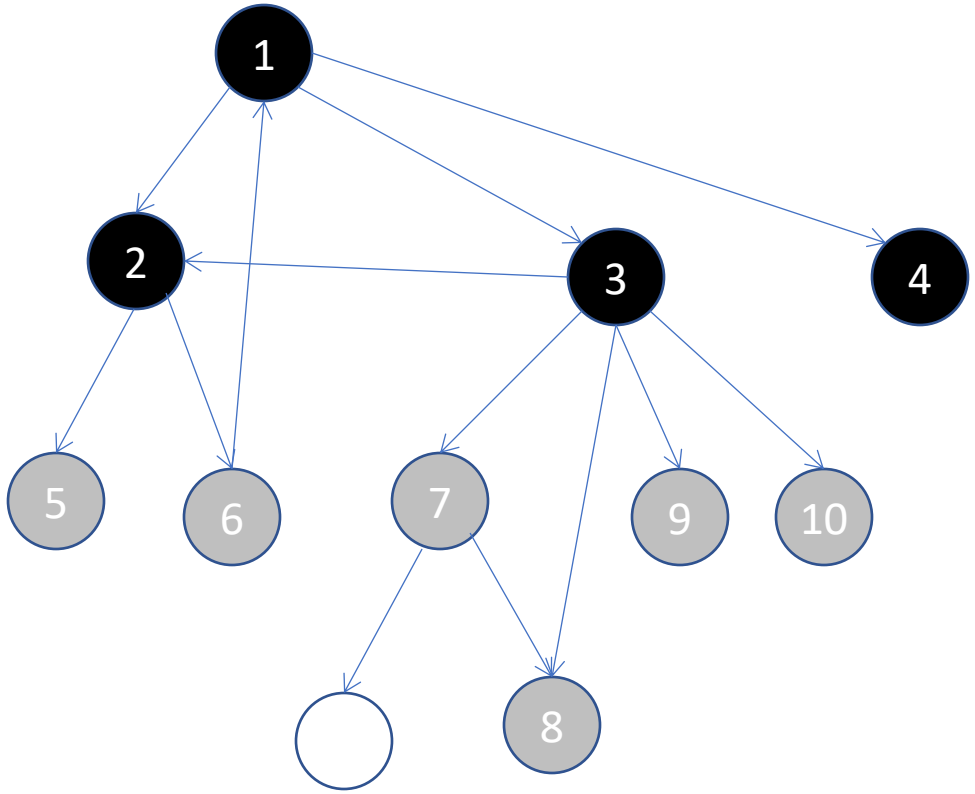



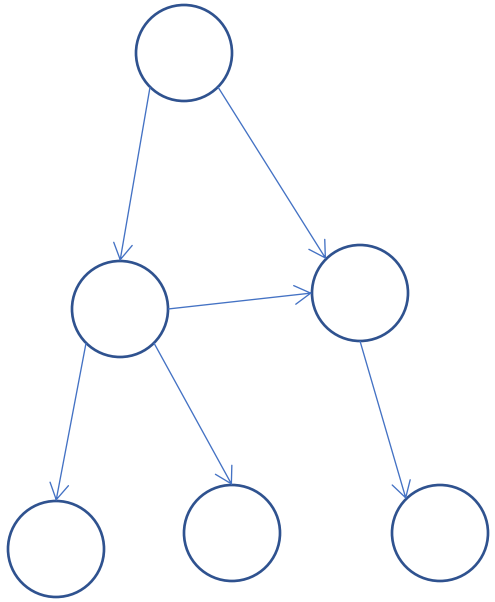
# BFS Traversal



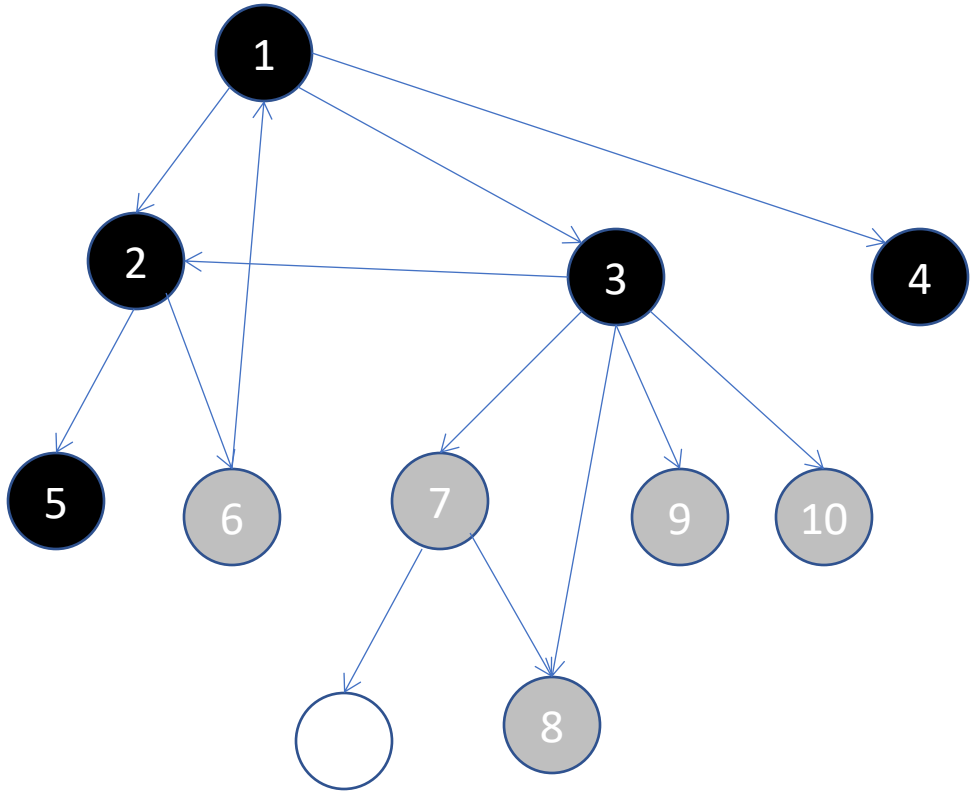



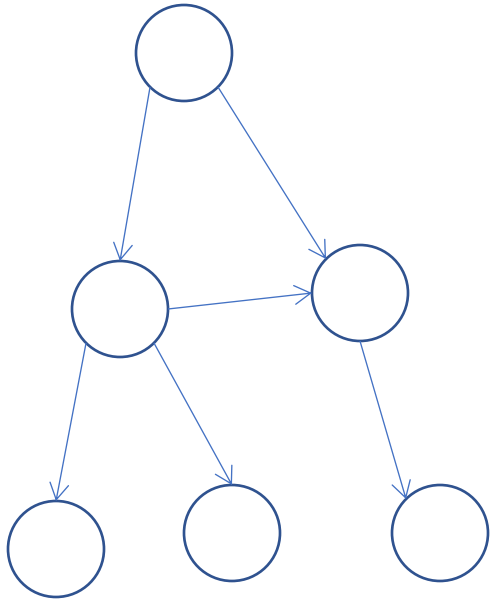
# BFS Traversal



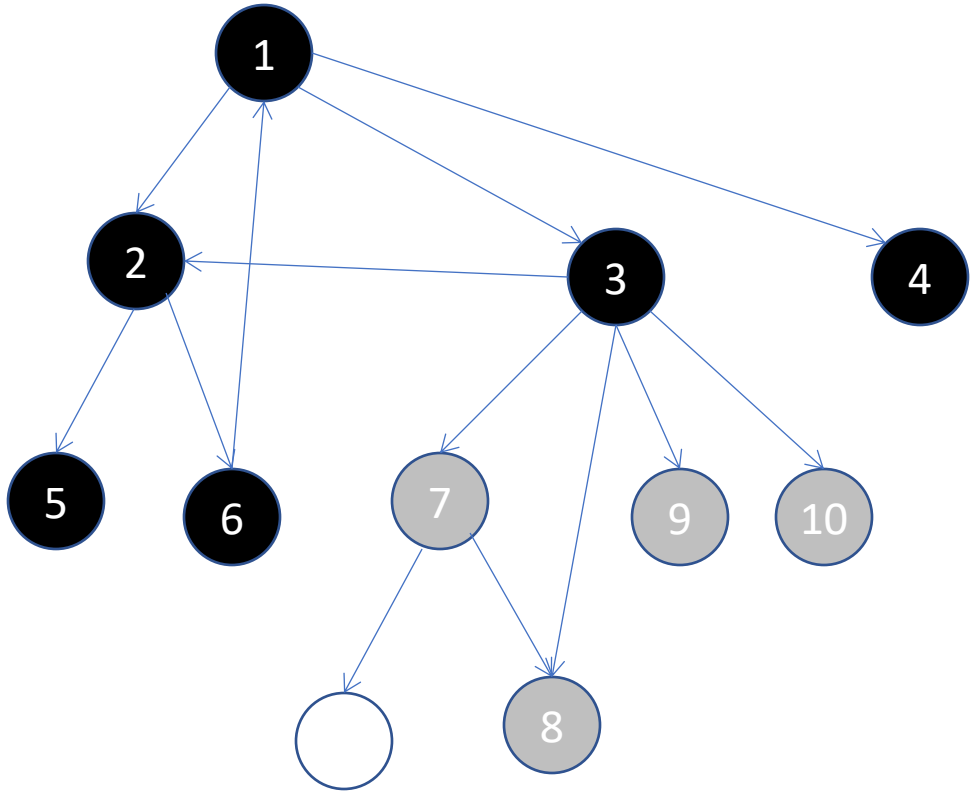



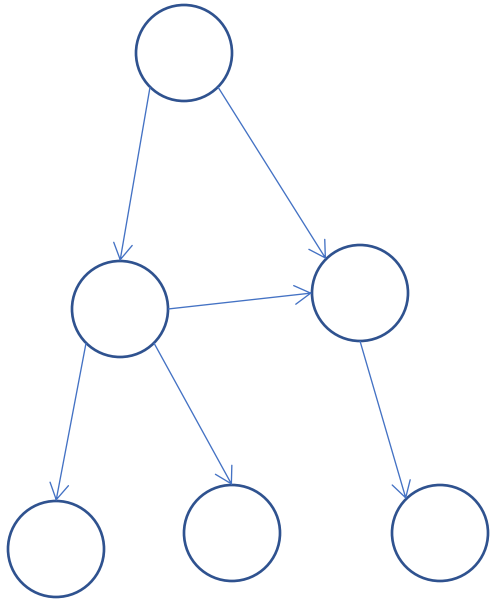
# BFS Traversal



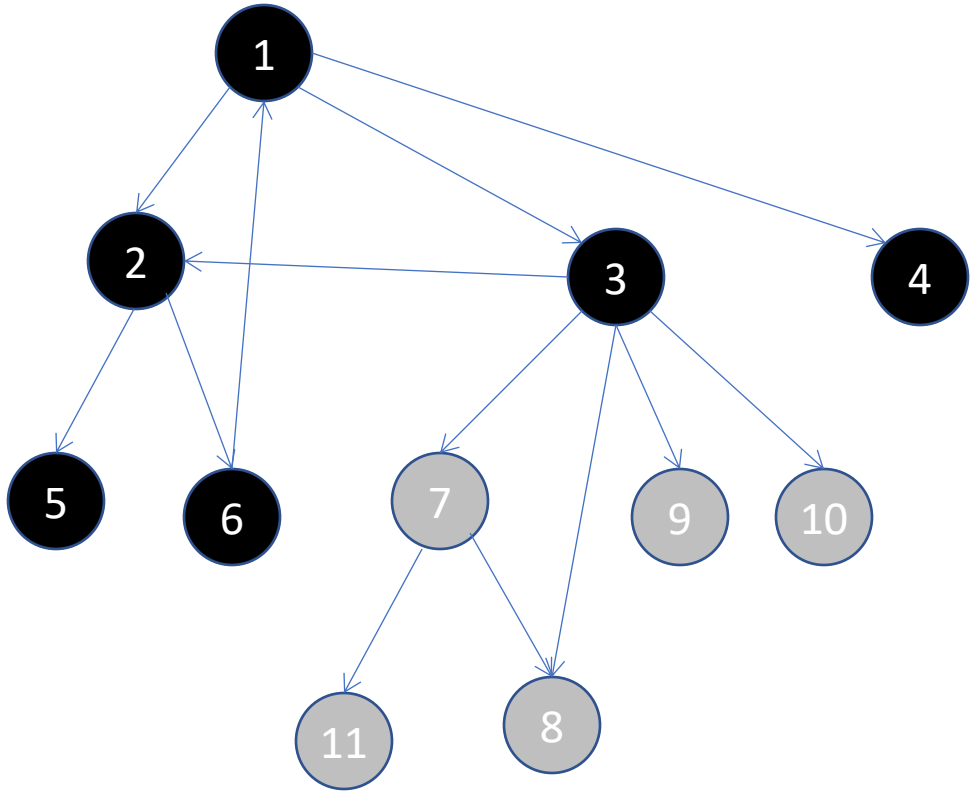



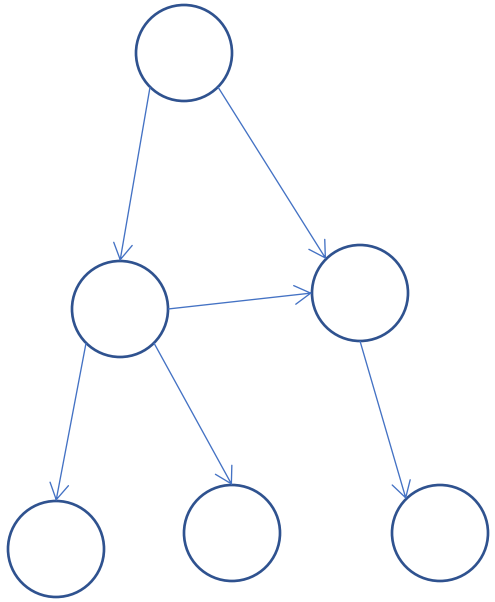
# BFS Traversal



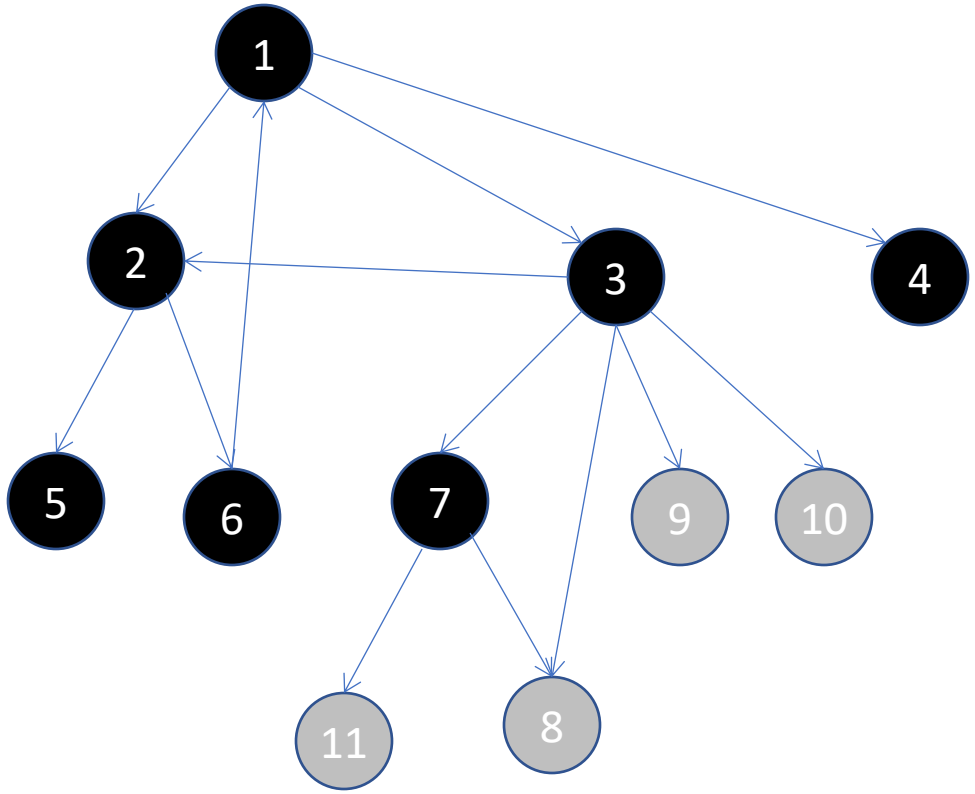



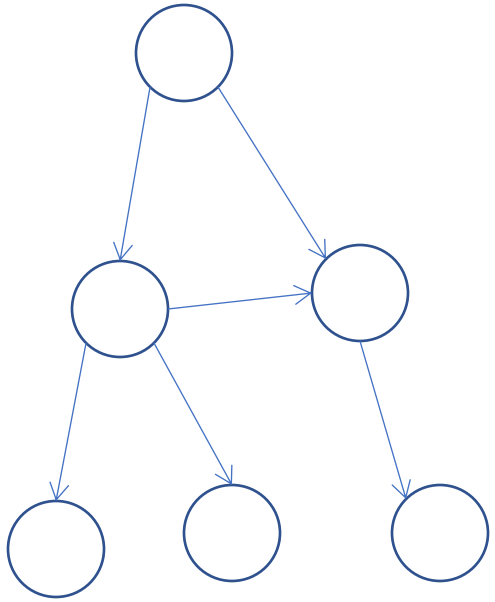
# BFS Traversal



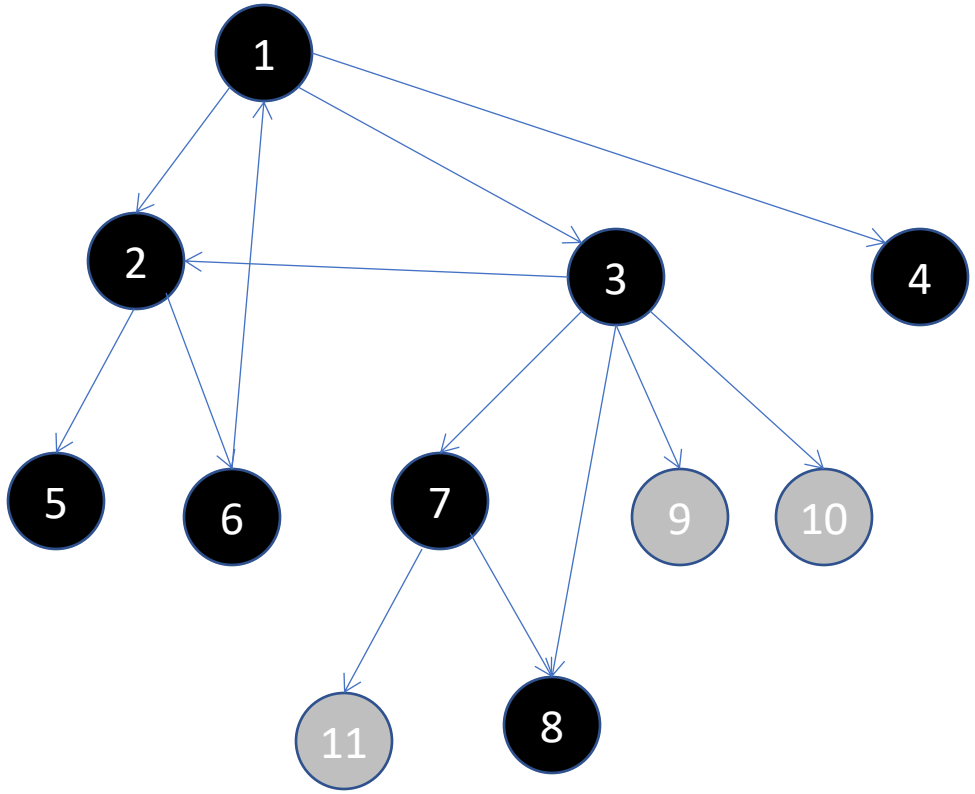



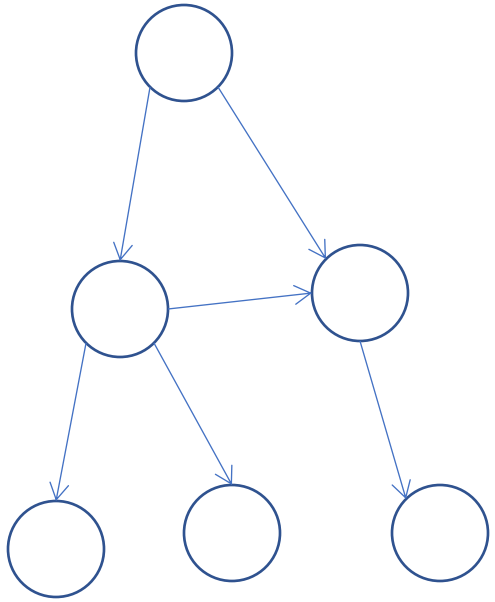
# BFS Traversal



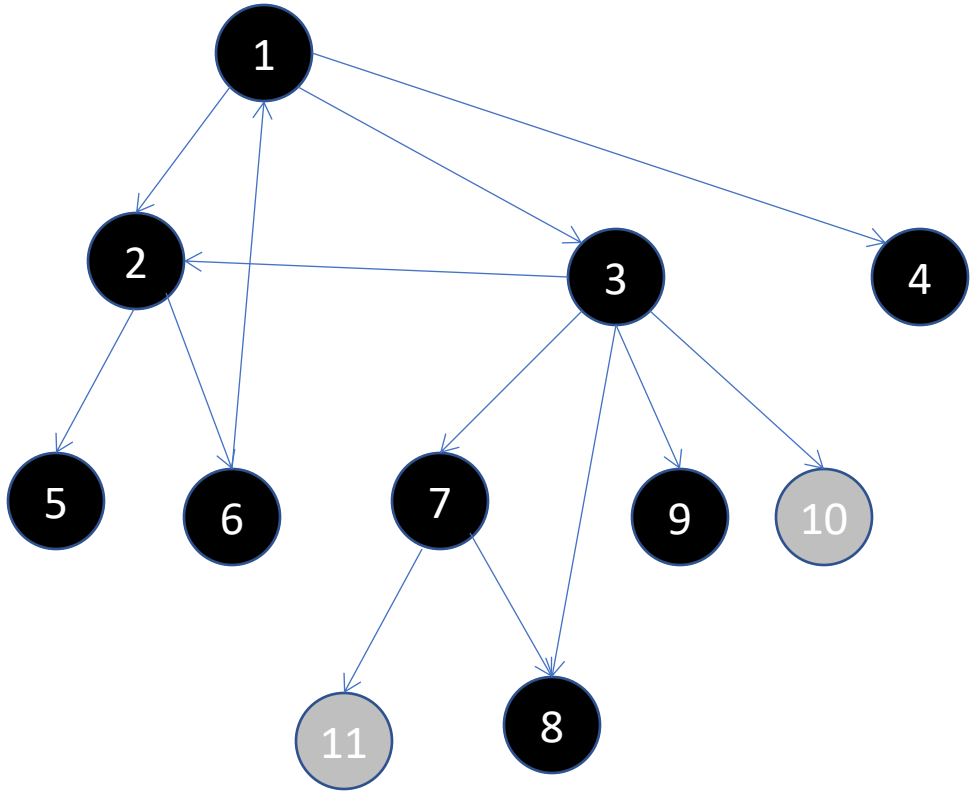



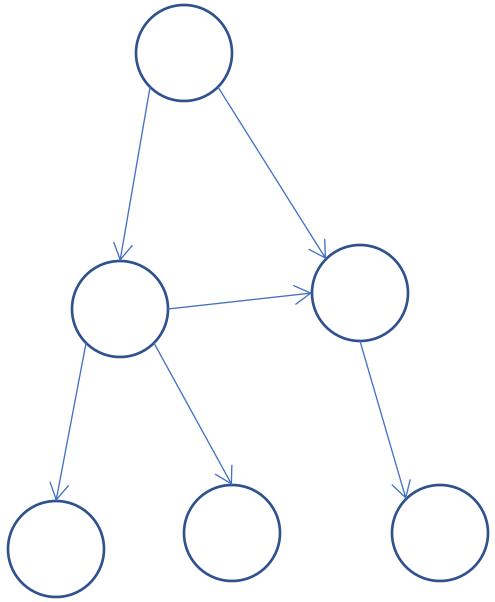
# BFS Traversal



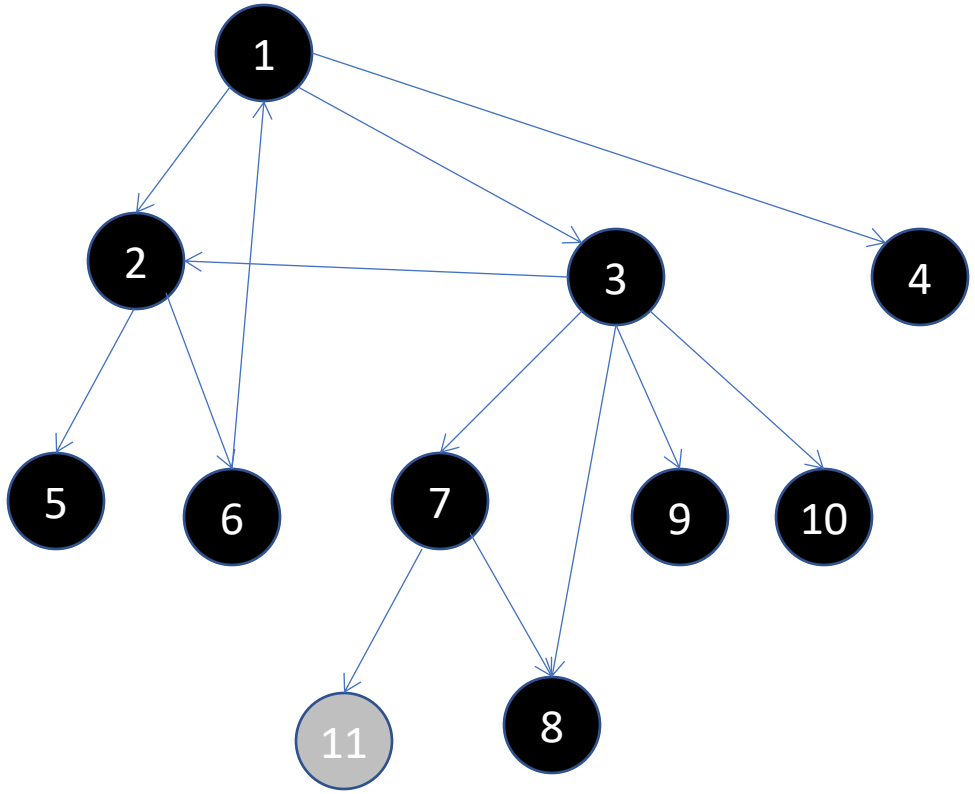



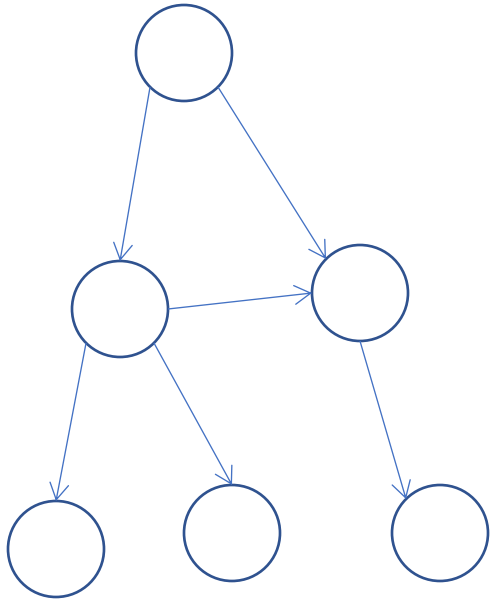
# BFS Traversal



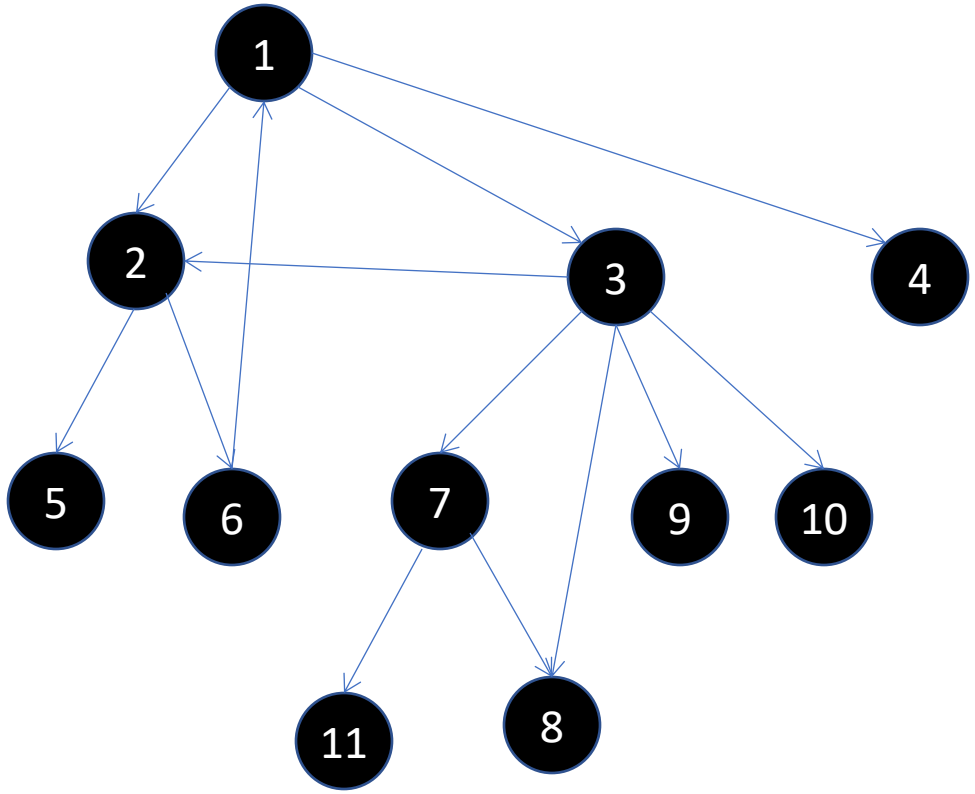



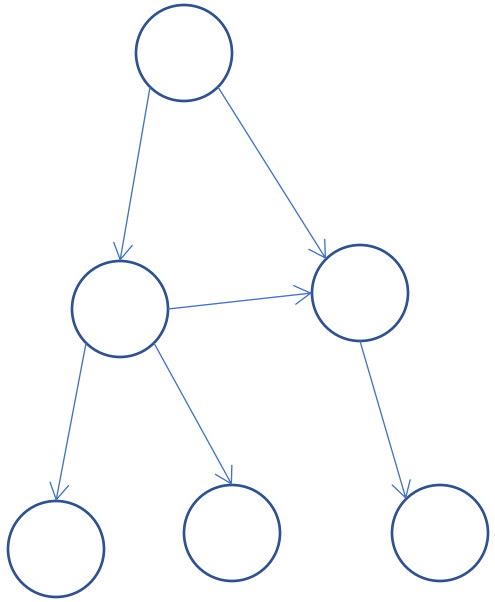
# BFS Traversal



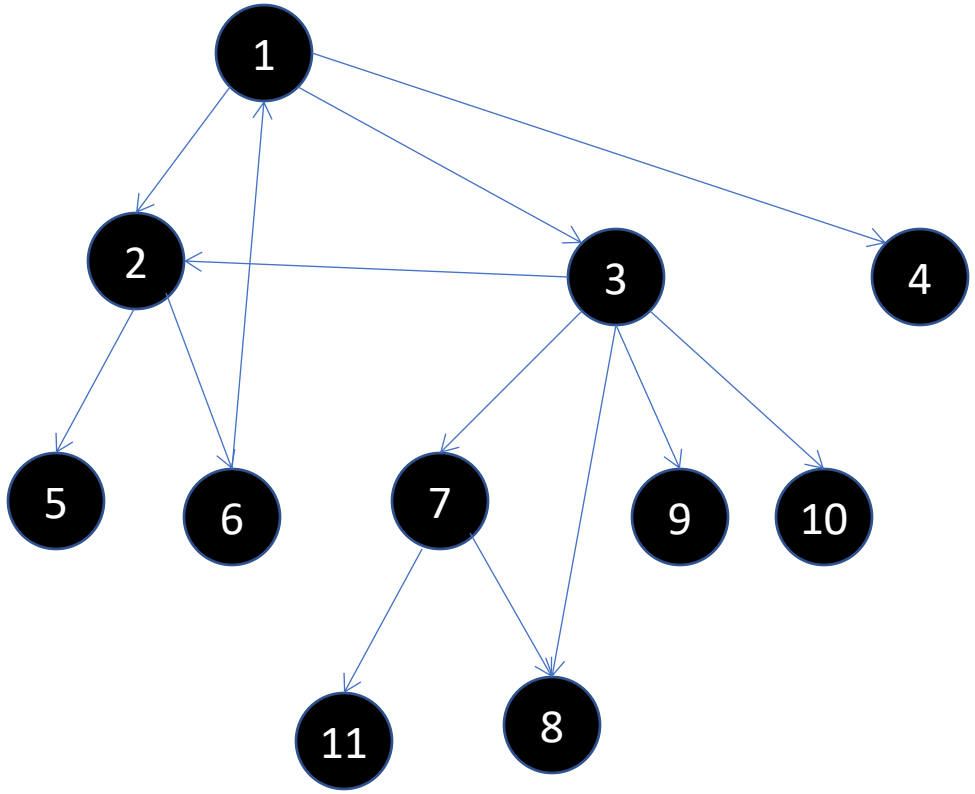



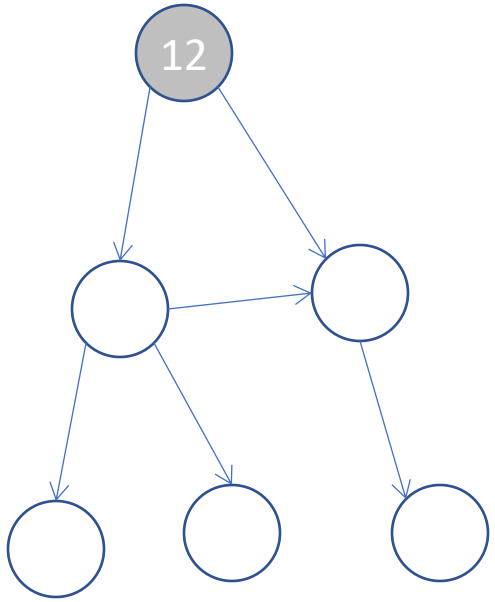
# BFS Traversal



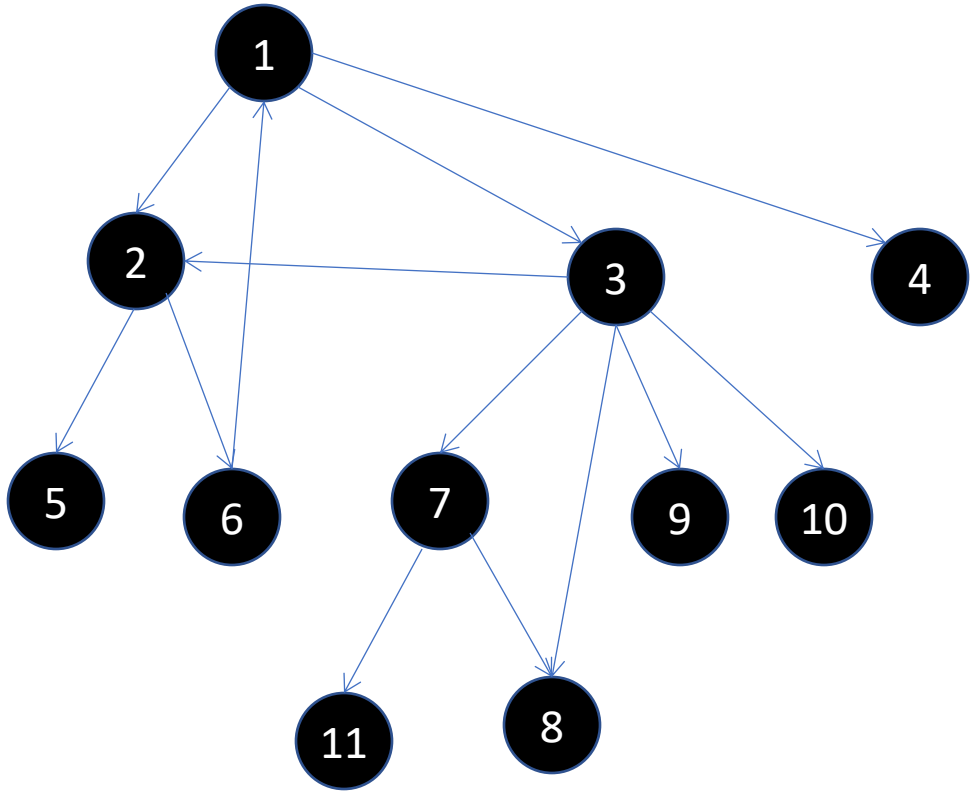



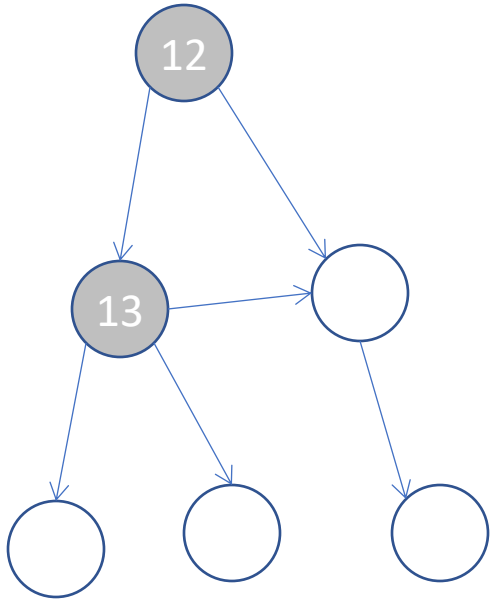
# BFS Traversal



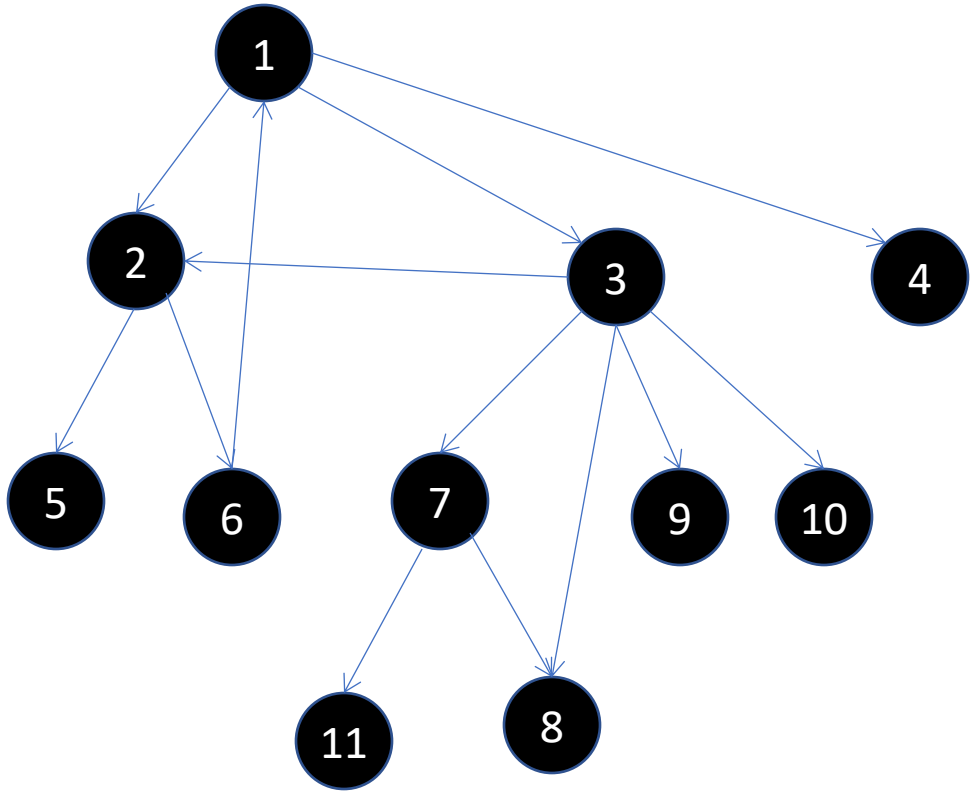



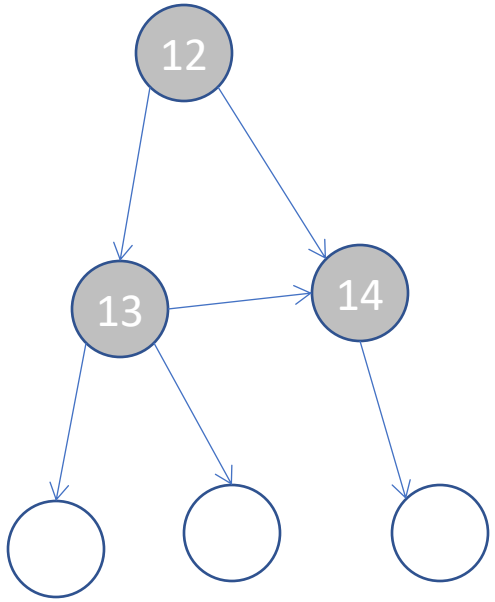
# BFS Traversal



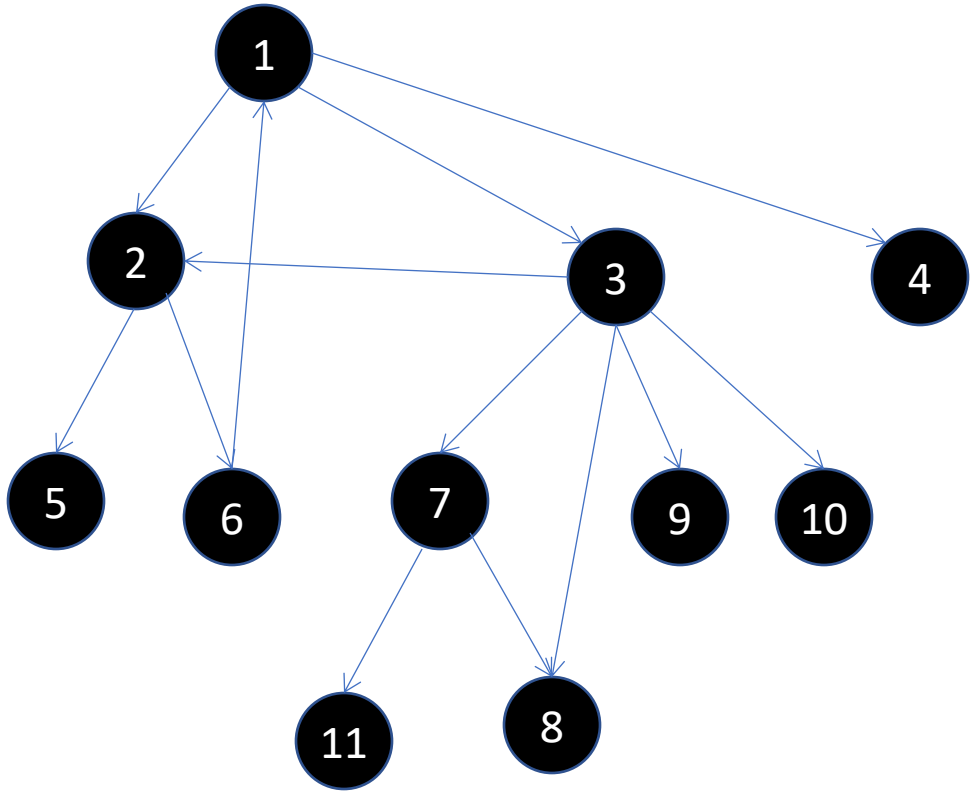



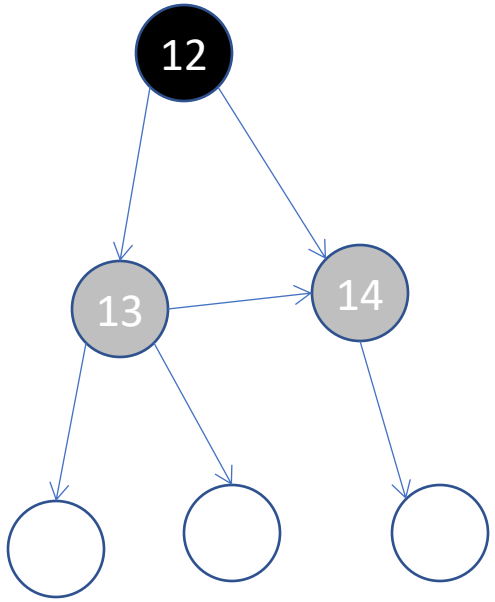
# BFS Traversal



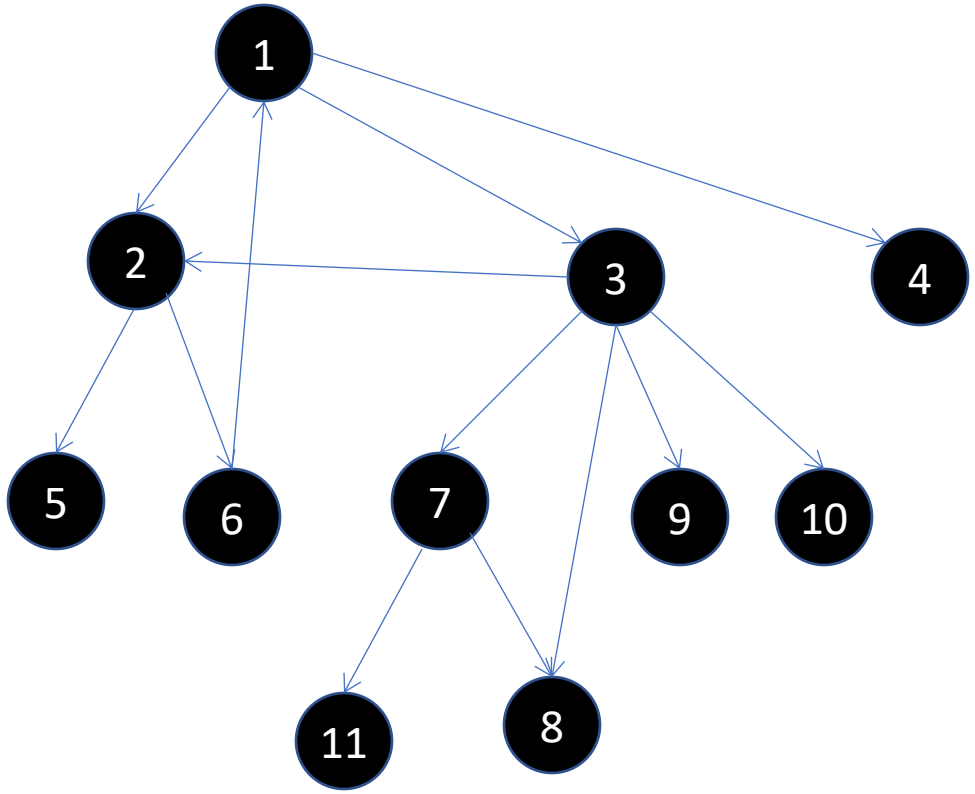



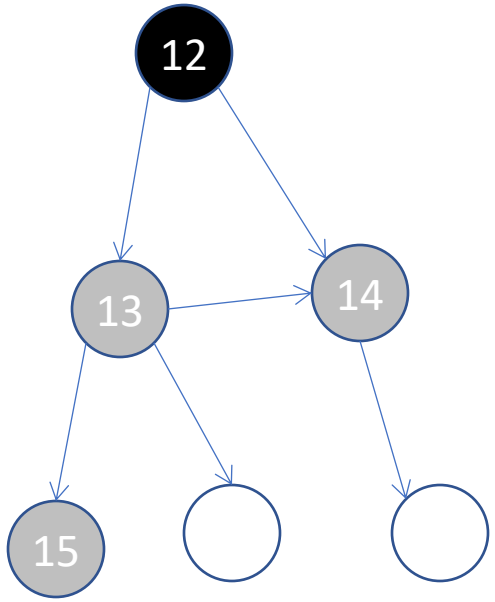
# BFS Traversal



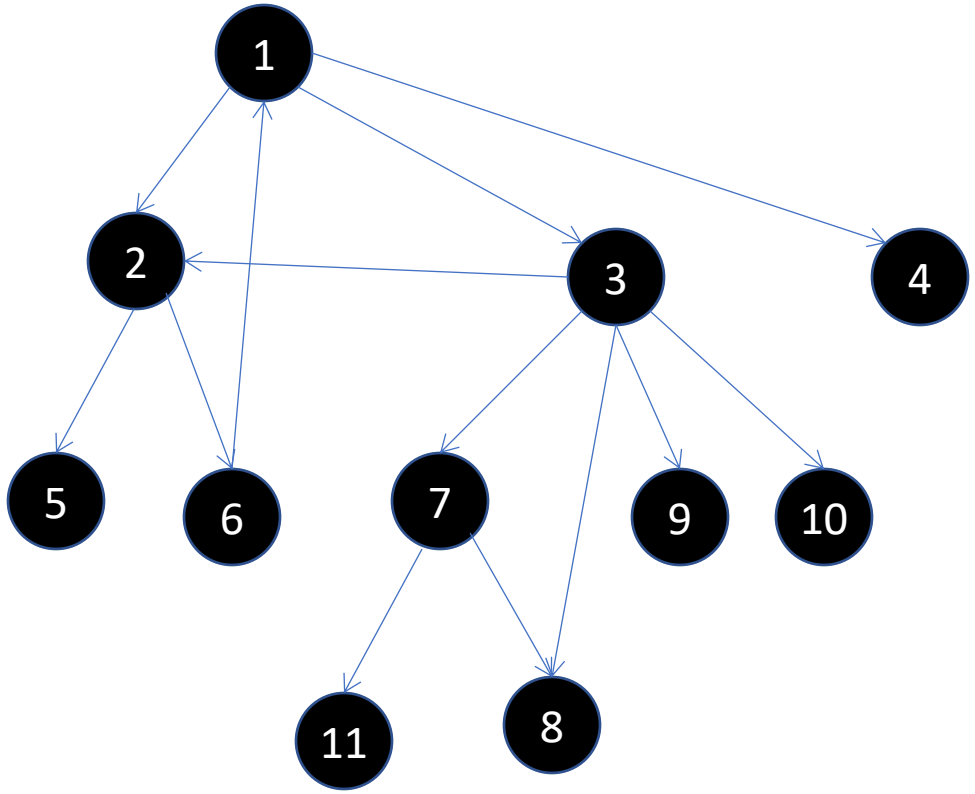



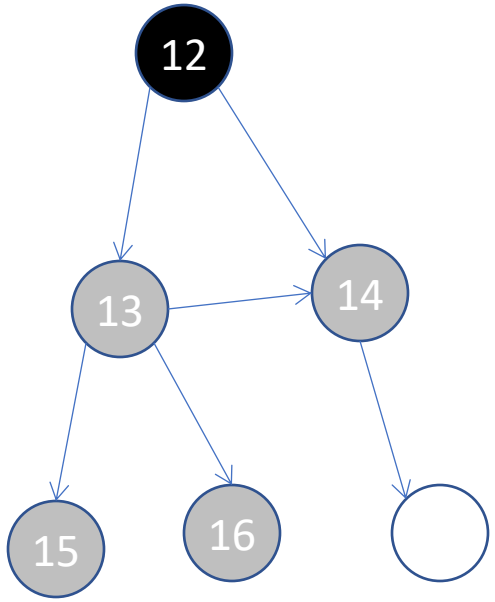
# BFS Traversal



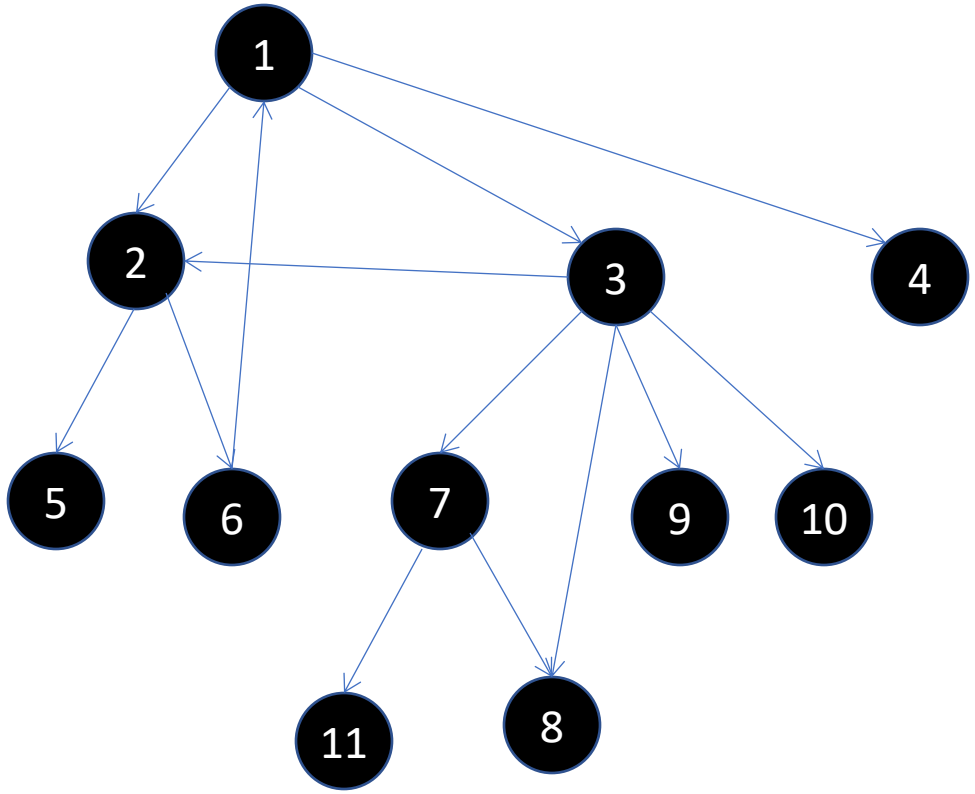



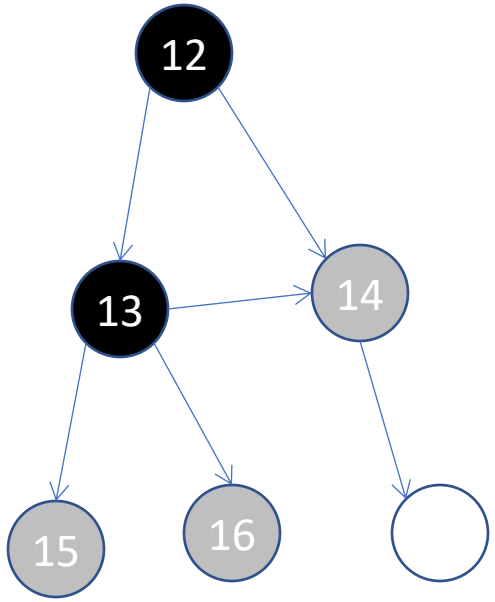
# BFS Traversal



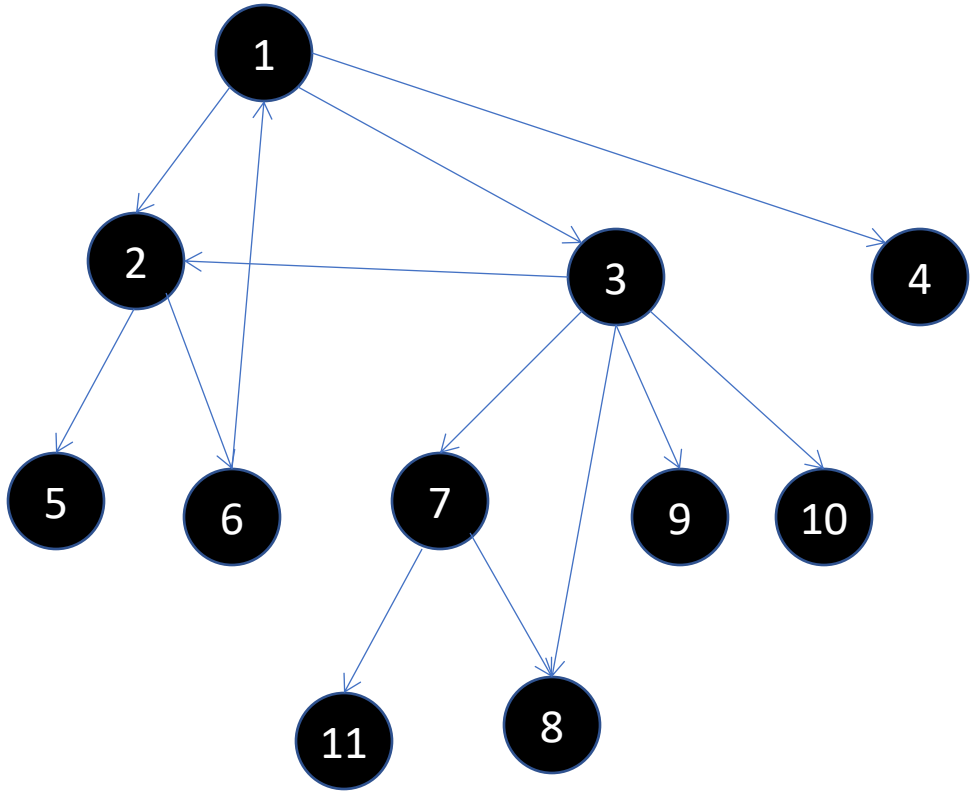



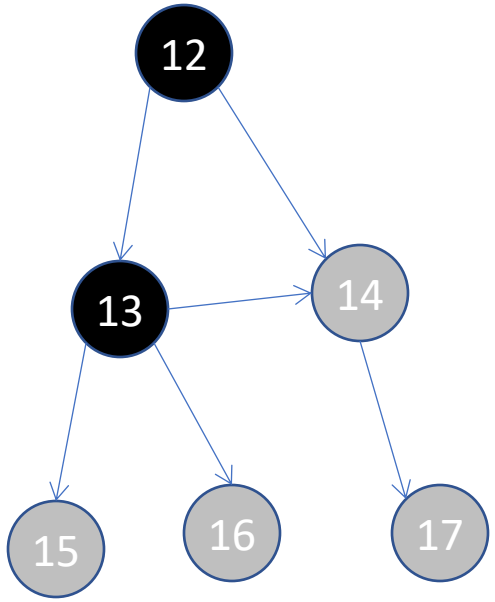
# BFS Traversal



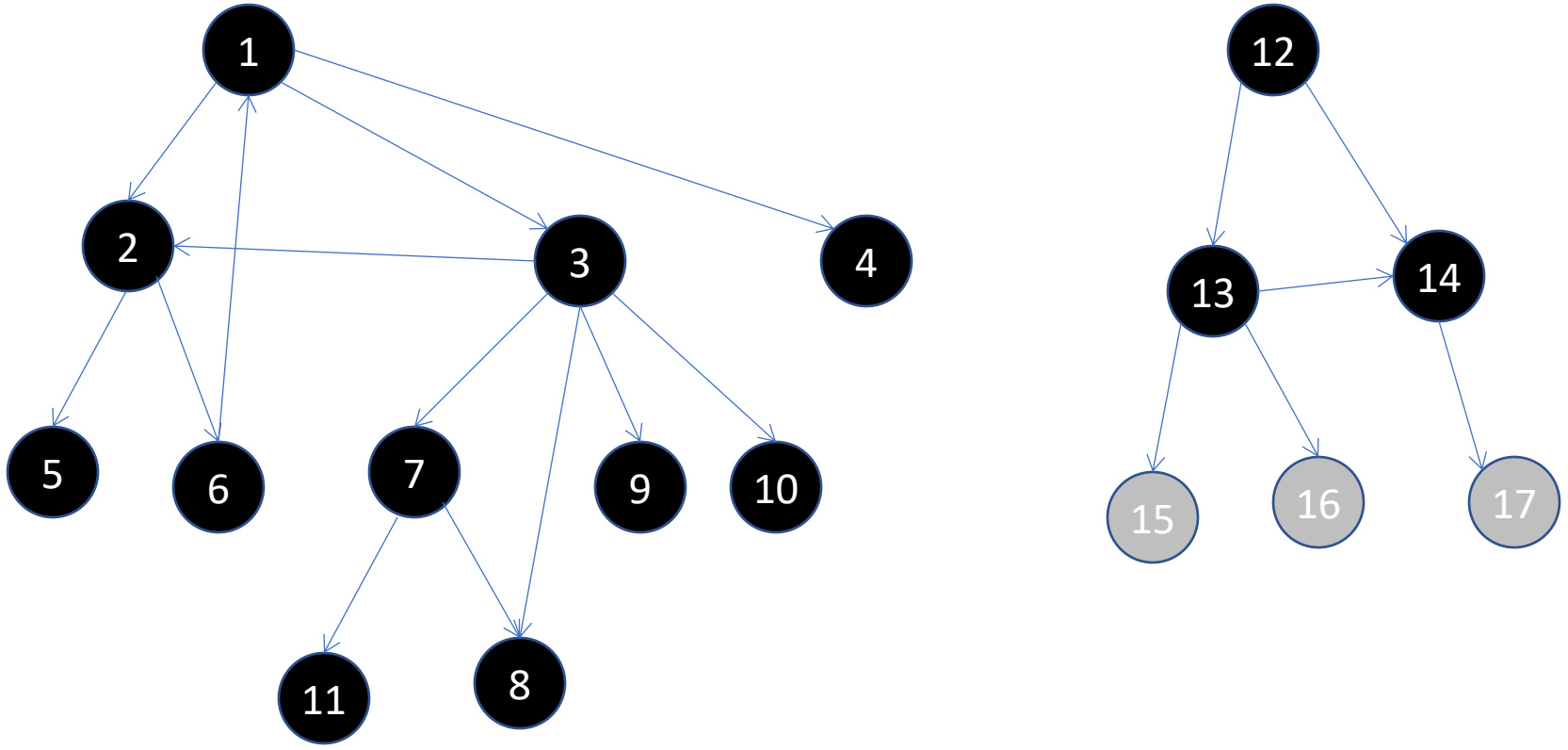



# BFS Traversal

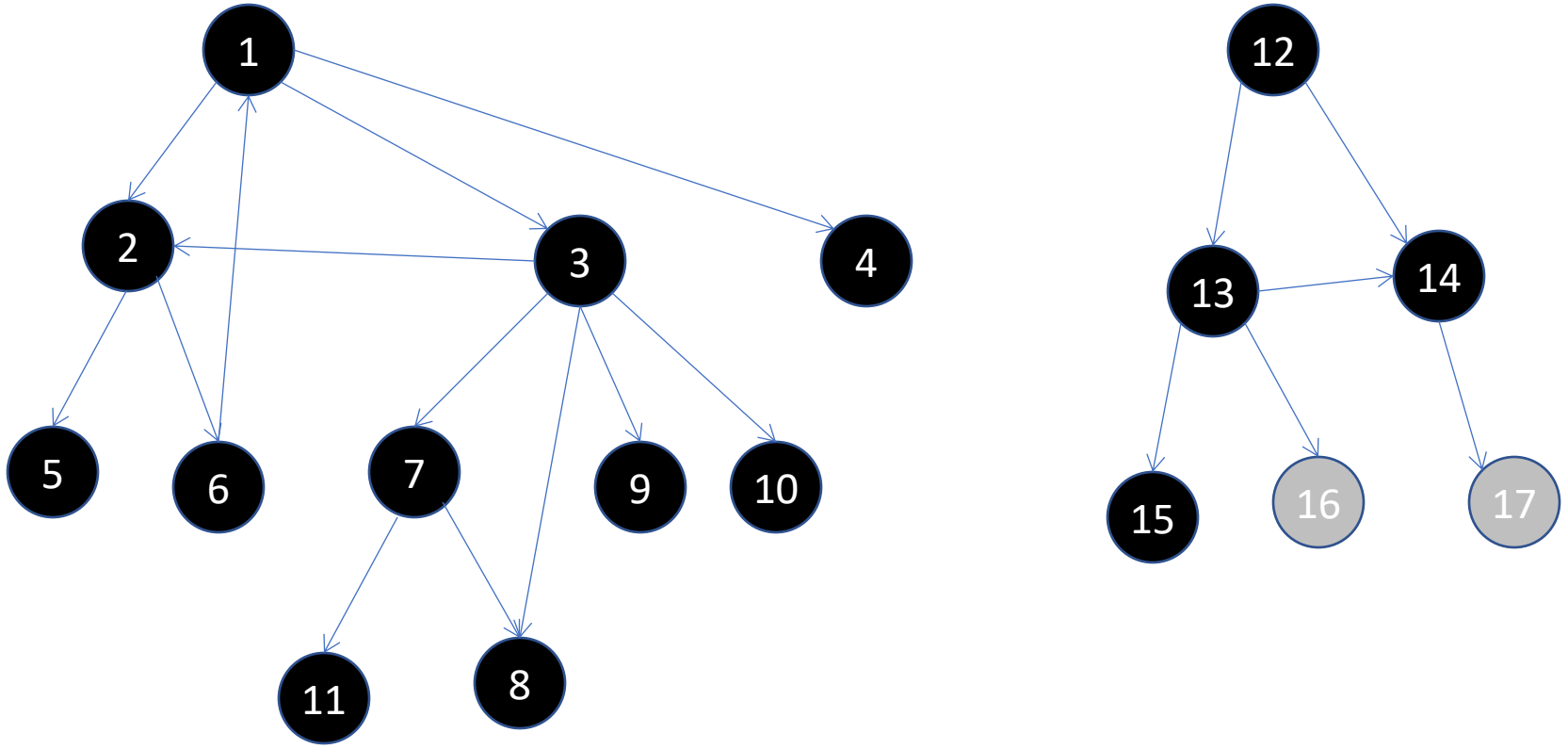


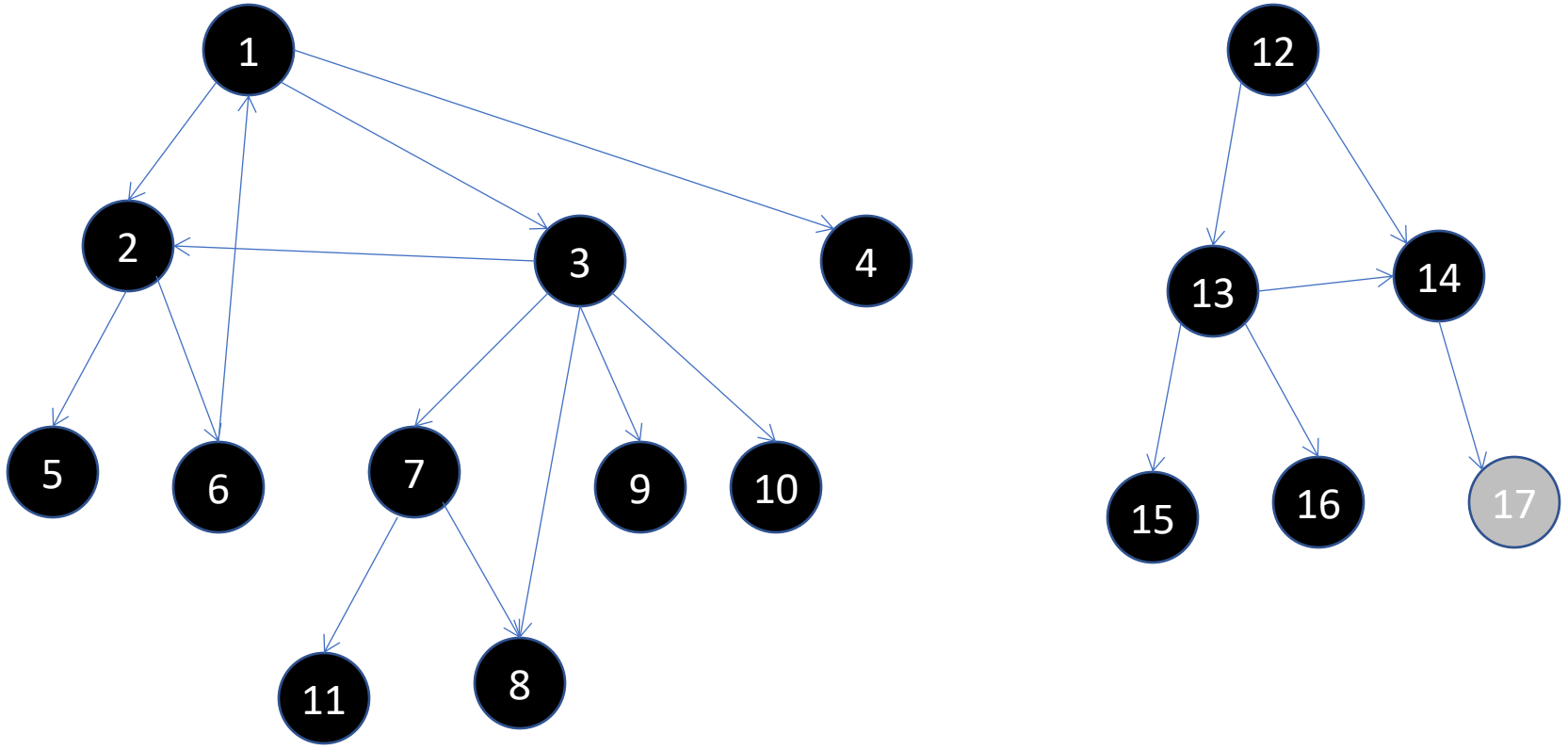
# BFS Traversal



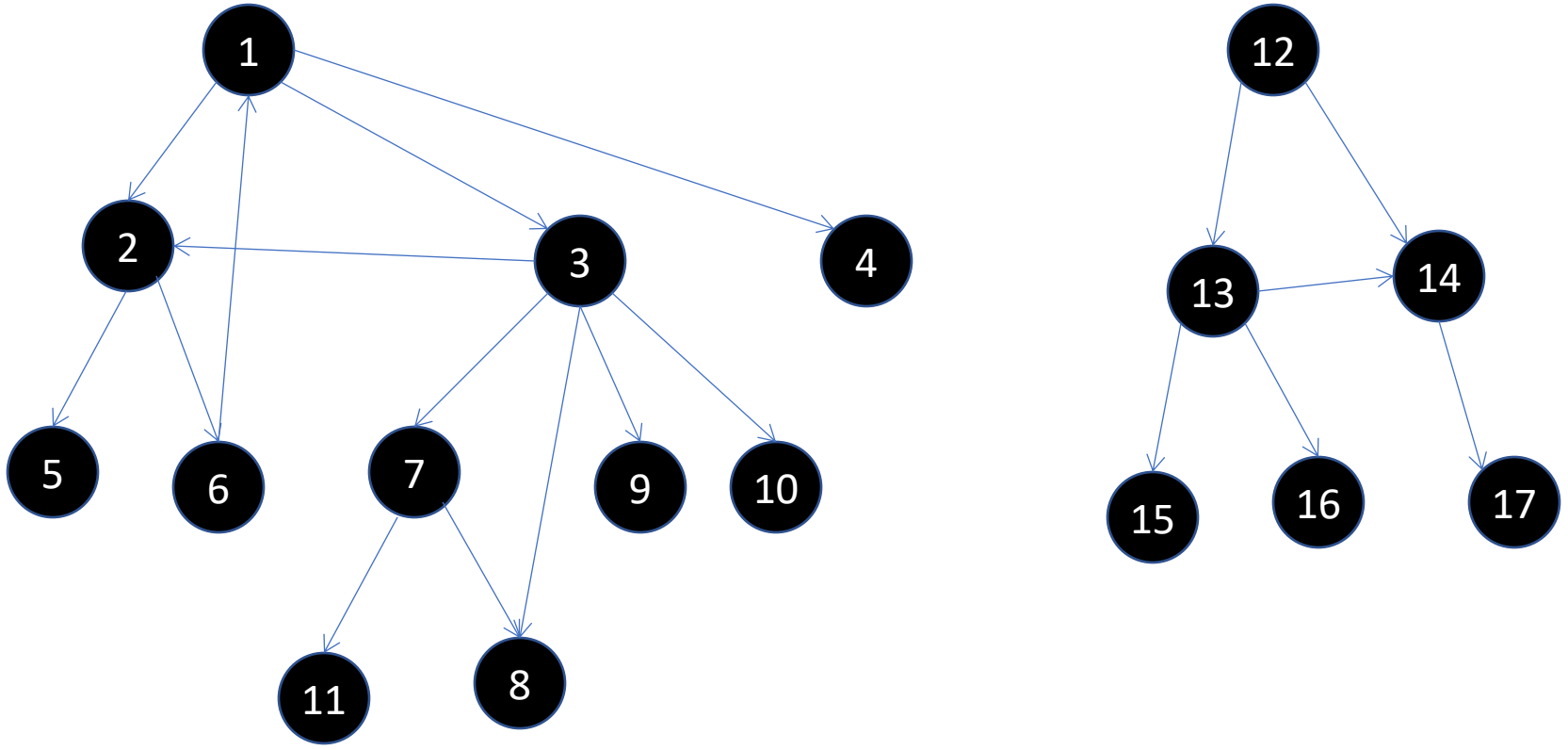
# BFS Traversal



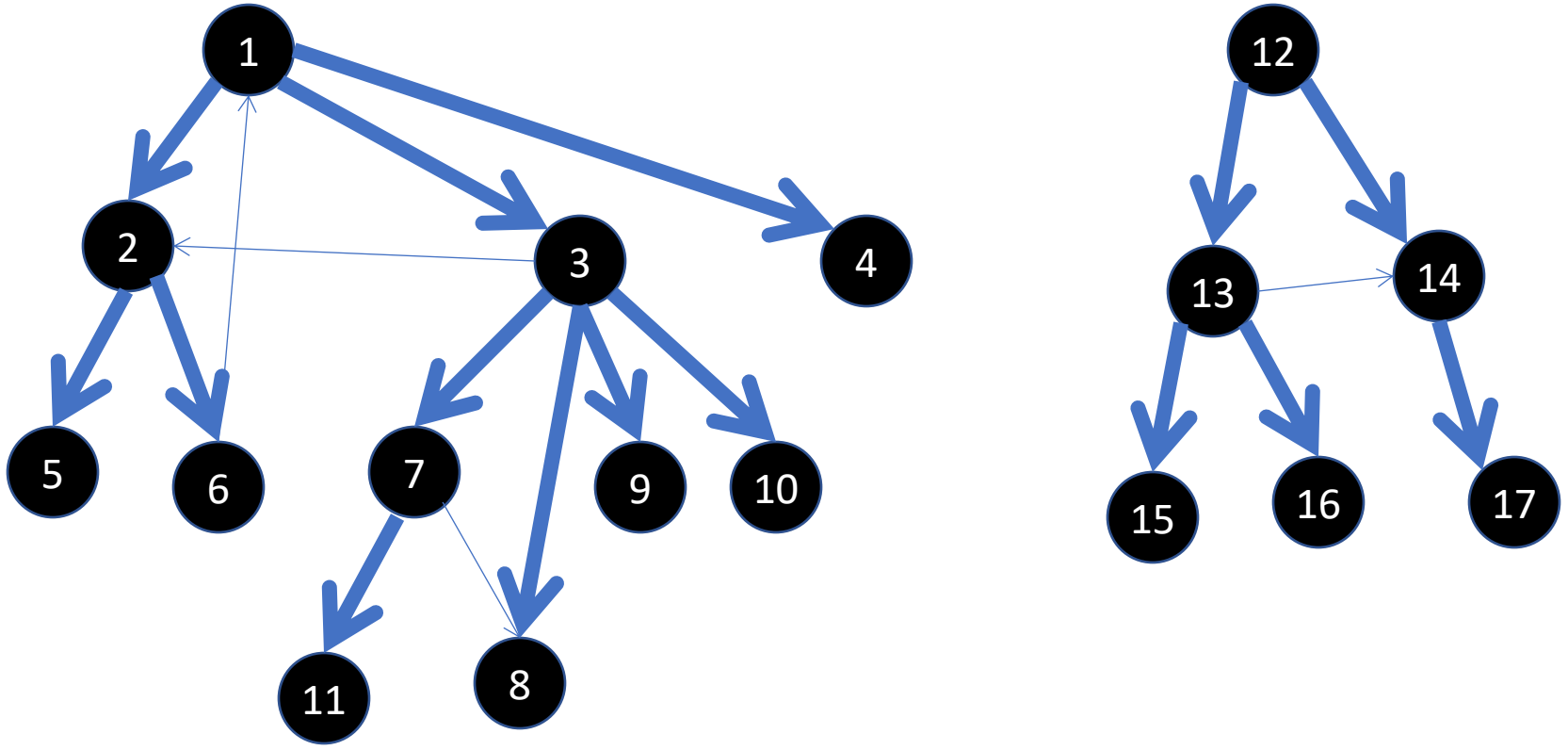
# BFS Traversal



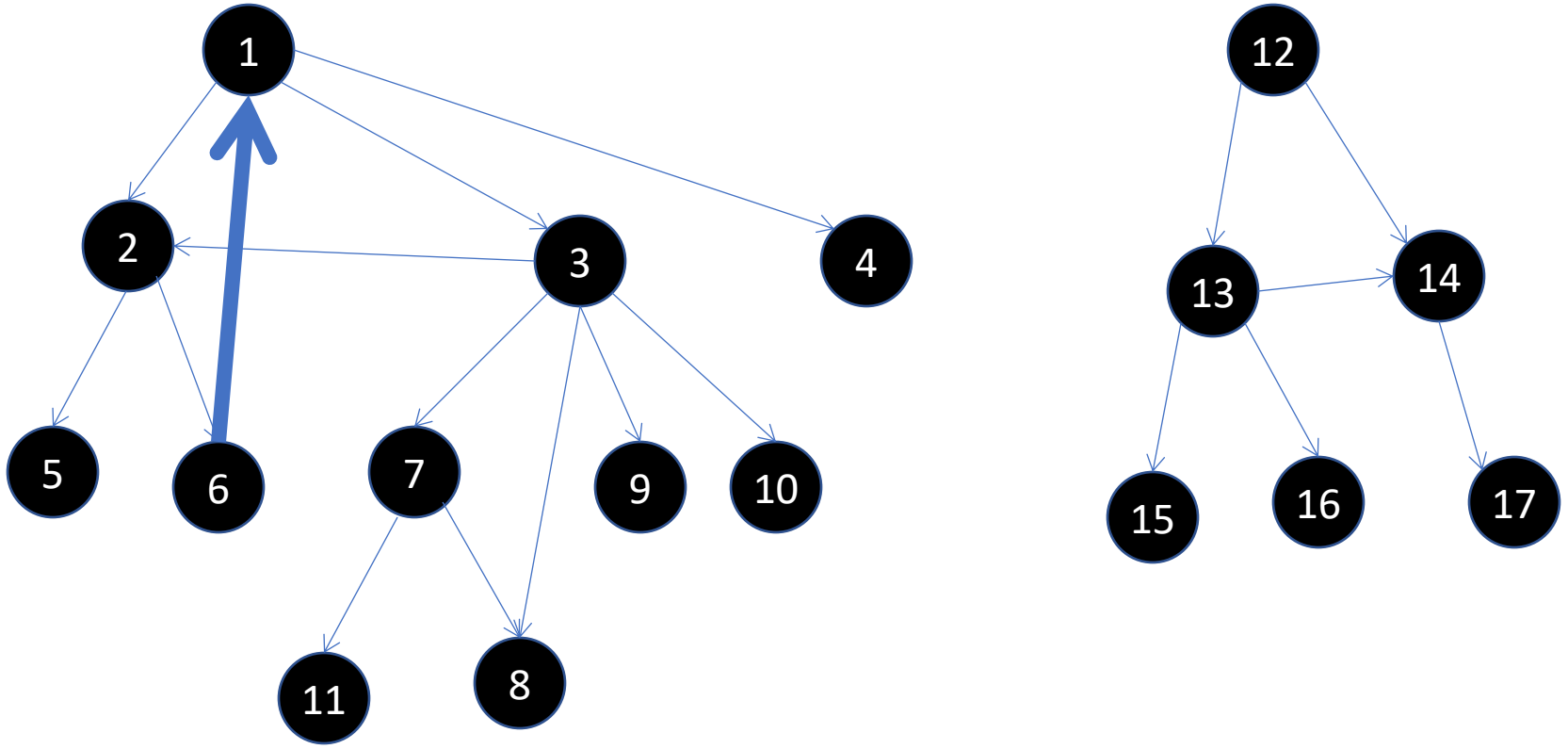
# BFS Traversal



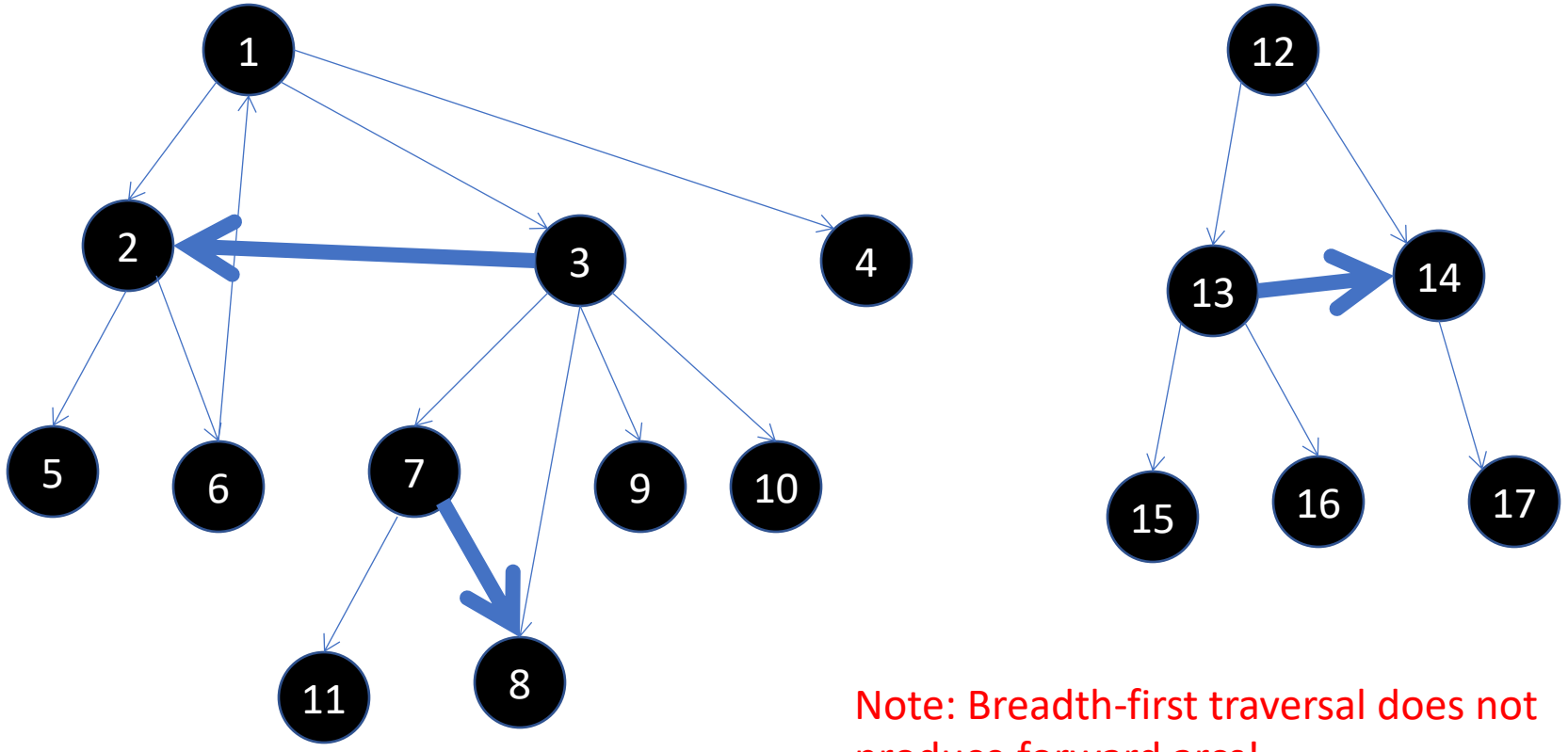
# BFS Traversal: Tree Arcs



# BFS Traversal: Back Arc



# BFS Traversal: Cross Arcs




Note: Breadth-first traversal does not produce forward arcs!

# Data Structure: DFS + Stacks

- A depth-first traversal operates like a stack (last-in-first-out, LIFO)
- Vertices are pushed onto the stack when we colour them grey
- We always operate on the vertex that is on top of the stack
- If the vertex on top of the stack has an (out-)neighbour that is still white, we push that (out-)neighbour onto the stack (we colour it grey)
- If the vertex on top of the stack has no more white (out-)neighbours, we pop the vertex off the stack (we colour it black)
- Depth-first traversals are usually implemented via a recursive method/ function call (see textbook). By the way: When you see a stack mentioned anywhere, the one word that should pop into your head is "recursive"…

# Data Structure: BFS + Queues

- A breadth-first traversal operates like a queue (first-in-first-out, FIFO)
- Vertices are added to the end of the queue and when we do we colour them grey
- We always operate on the vertex that is at the front of the queue (the vertex that has been grey for the longest time)
- If the vertex at the front of the queue has an (out-)neighbour that is still white, we add that (out-)neighbour to the end of the queue (we colour it grey)
- If the vertex at the front of the queue has no more white (out-)neighbours, we take the vertex out of the queue (we colour it black)

# Time Complexity: DFS v.s. BFS

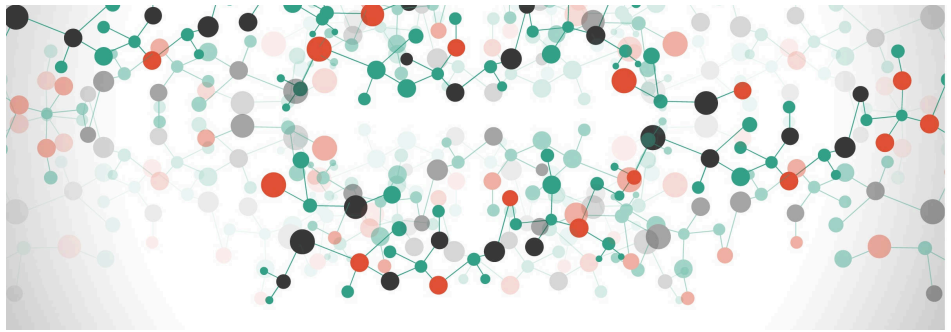
- Both BFS and DFS have same complexity
  - the next grey/white node has constant complexity

- With adjacency matrix: Need to find out-neighbours for $n$ vertices. With $\Theta(n)$ per vertex, this means $\Theta(n^2)$.

- With adjacency list: the sequence for each node is only visited once and we need to visit all the edges. So the complexity is $\Theta(n+e)$.

# OUTLINE

- Graph Traversal Algorithms
  - Depth-first Search (DFS)
  - Breadth-first Search (BFS)
  - Priority-first Search (PFS)
- Implementation
  - Stack – DFS
  - Queue – BFS
  - Priority Queues – PFS



# Priority-first Search (PFS)

- In PFS, the next grey vertex is selected according to a priority value
- Typically, this is an integer, such that the grey vertex with the lowest value is selected first
- Priority value is assigned (often computed) no later than when the vertex turns grey

# Priority-first Search (PFS)

- In simple PFS, the priority value of a vertex does not change.
- In advanced PFS, the priority value of a vertex may be updated again later.

- BFS and DFS are really just special cases of simple PFS.
  - In BFS, the priority values are the order in which the vertices turn grey (1, 2, 3, …).
  - In DFS, the priority values are the negative order in which the vertices turn grey (-1, -2, -3, …).

# Time Complexity: PFS

- General time complexity is not very good: Need to find minimum priority value each time we need to select the next grey node
- If we search up to $n$ vertices (say, in an array) for the minimum priority value, we need $\Omega(n^2)$
- This can be improved on a little by using a priority queue (e.g., a heap) for the grey vertices, but it's still slower than pure DFS or BFS
- Use PFS when there is an advantage in processing high priority vertices first (e.g., when this allows us to remove other vertices or edges from the (di)graph to be traversed, thereby reducing the search space)

# SUMMARY

- Graph Traversal Algorithms
  - Depth-first Search (DFS)
  - Breadth-first Search (BFS)
  - Priority-first Search (PFS)
- Implementation
  - Stack – DFS
  - Queue – BFS
  - Priority Queues – PFS