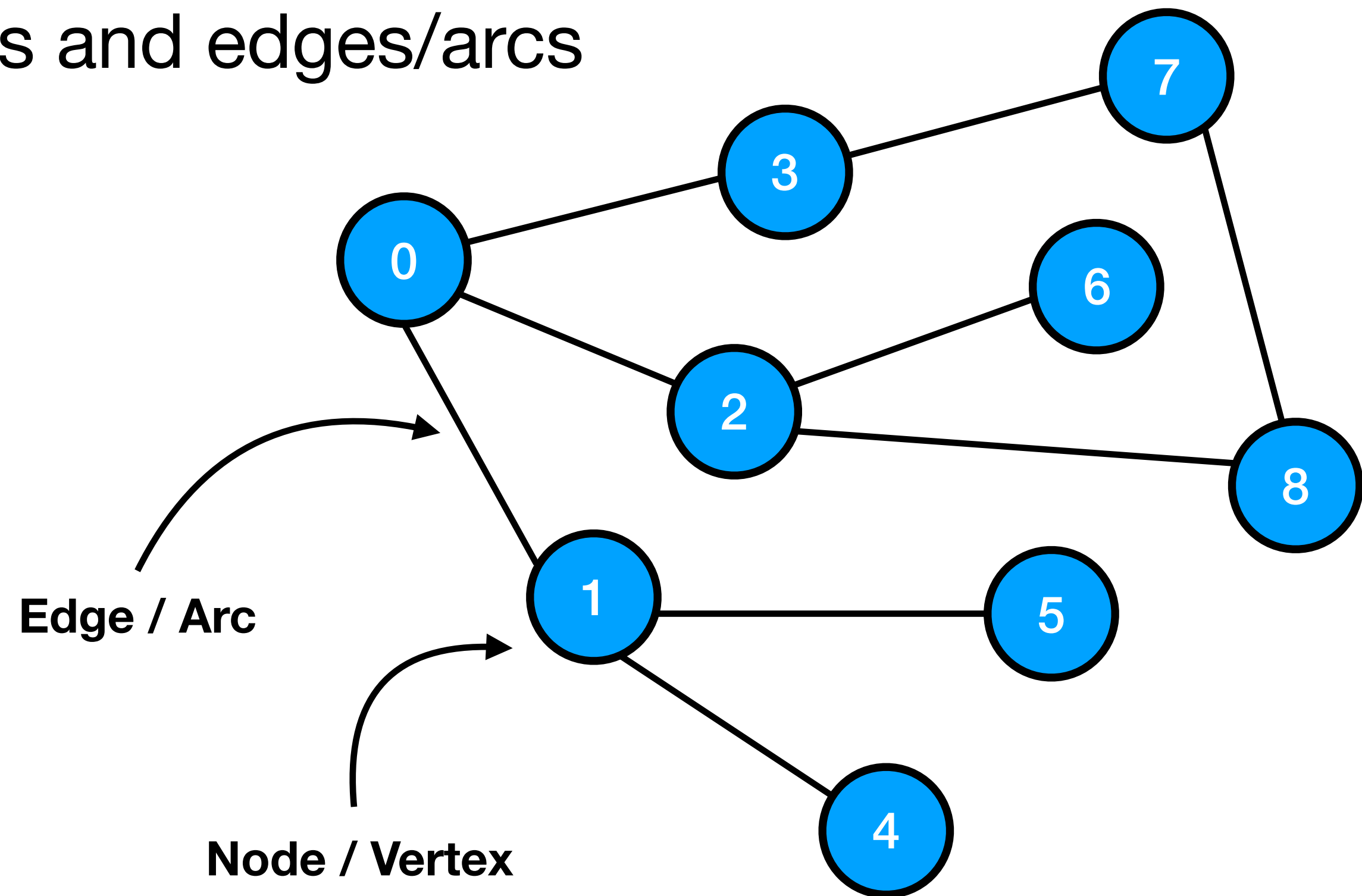


# Code Prime Theory

Graph Traversals

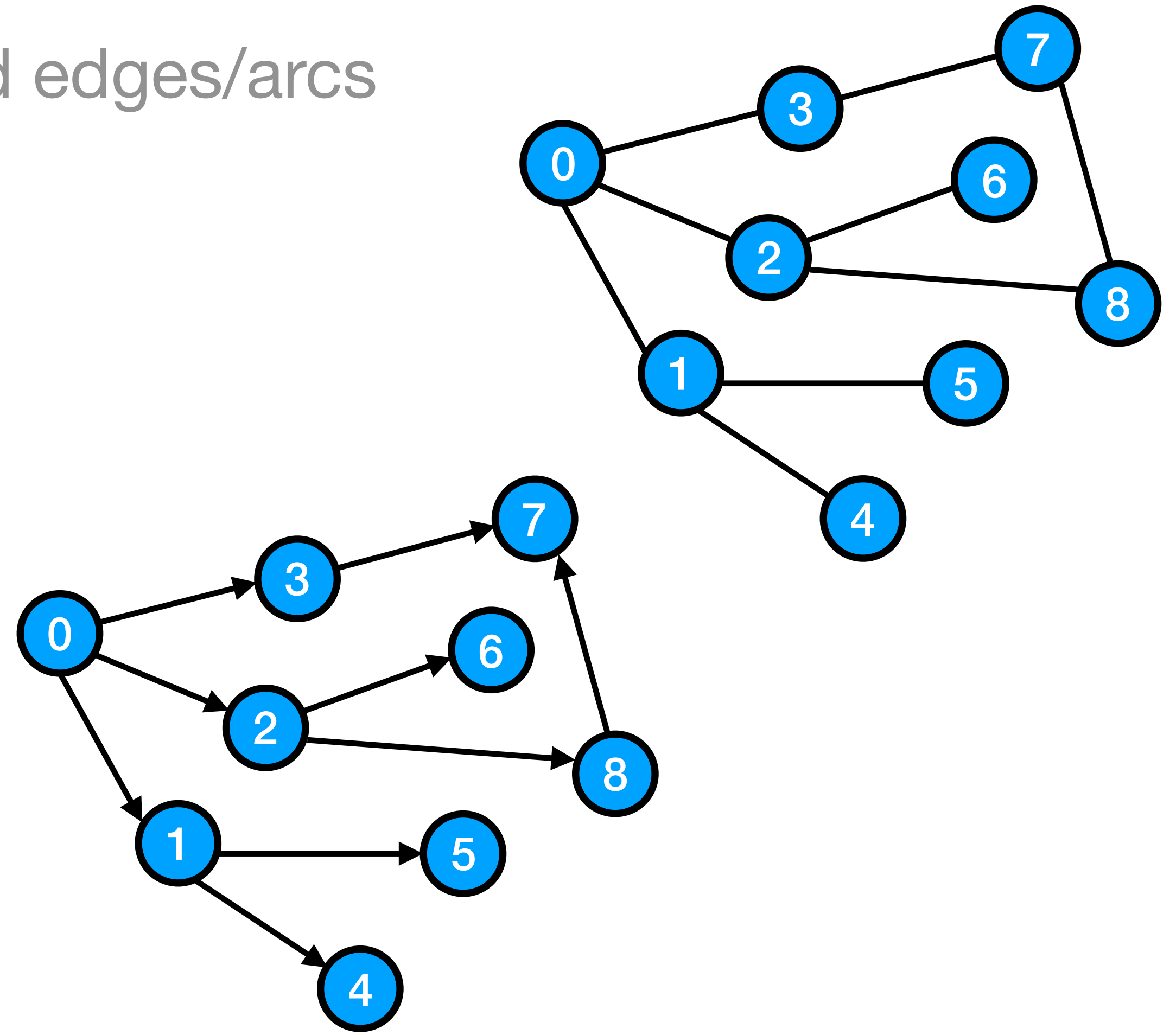
# Recap

- Graph consists of vertices/nodes and edges/arcs
- Undirected/Directed graph
- Weighted graph
- Graph representation



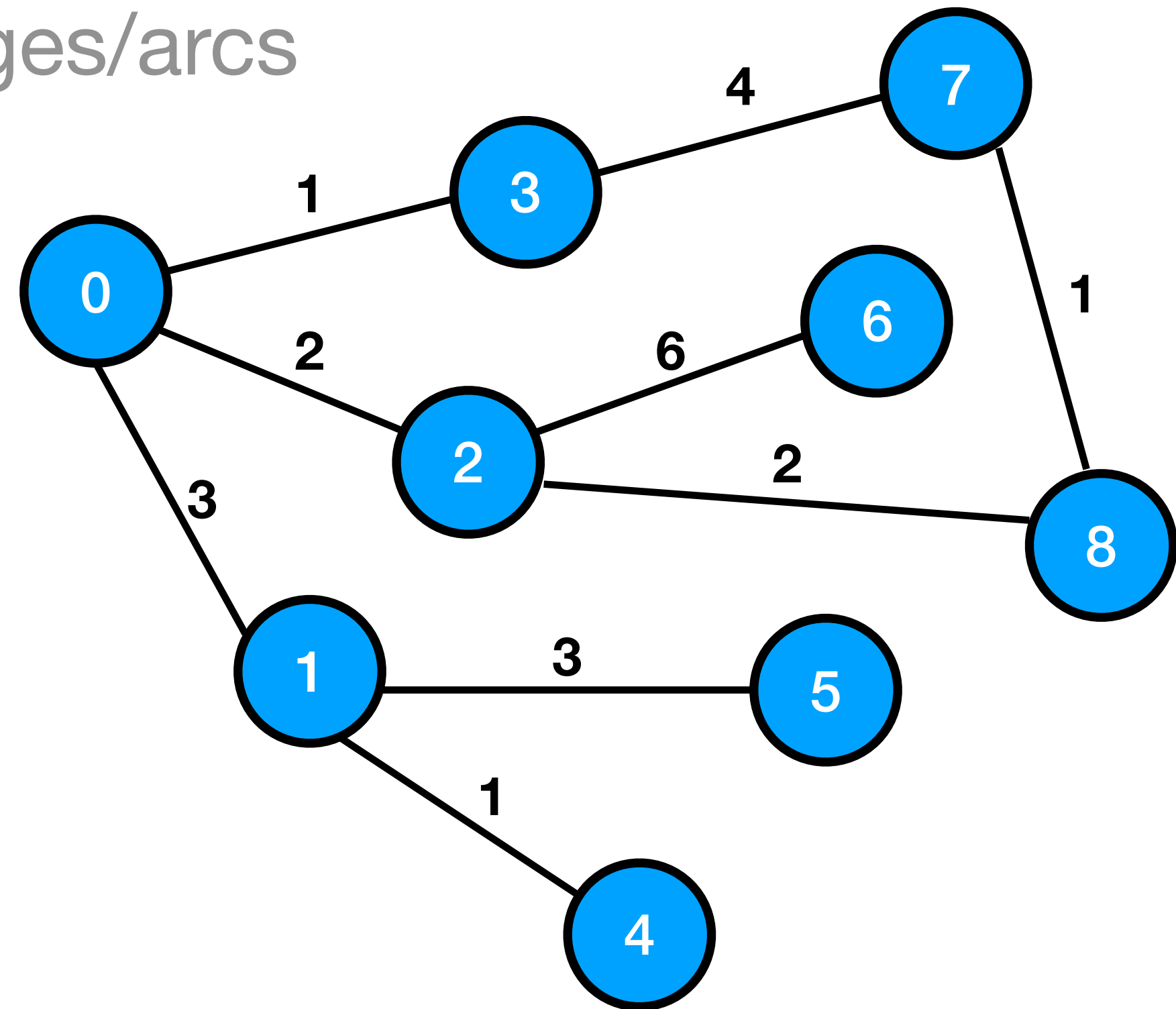
# Recap

- Graph consists of vertices/nodes and edges/arcs
- Undirected/Directed graph
- Weighted graph
- Graph representation



# Recap

- Graph consists of vertices/nodes and edges/arcs
- Undirected/Directed graph
- Weighted graph
- Graph representation

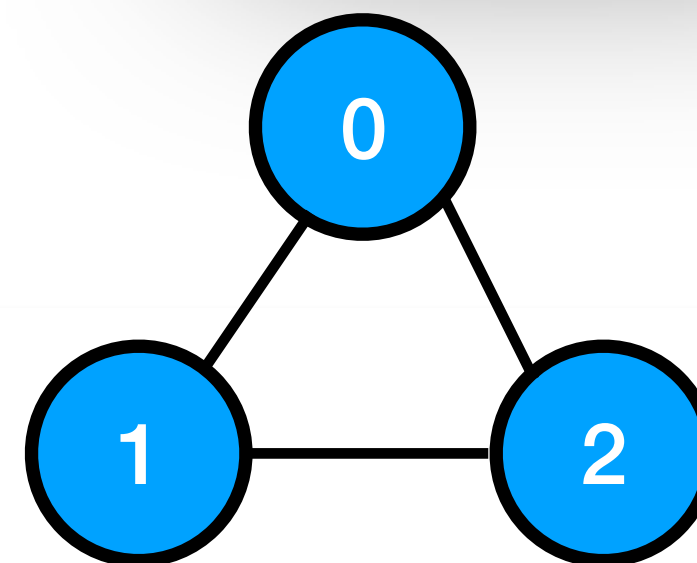


# Recap

- Graph consists of vertices/nodes and edges/arcs
- Undirected/Directed graph
- Weighted graph
- Graph representation

```
Edge list
[
  [0, 1],
  [1, 2],
  [2, 0]
]
```

```
Adjacency matrix
[
  [0, 1, 1],
  [1, 0, 1],
  [1, 1, 0]
]
```



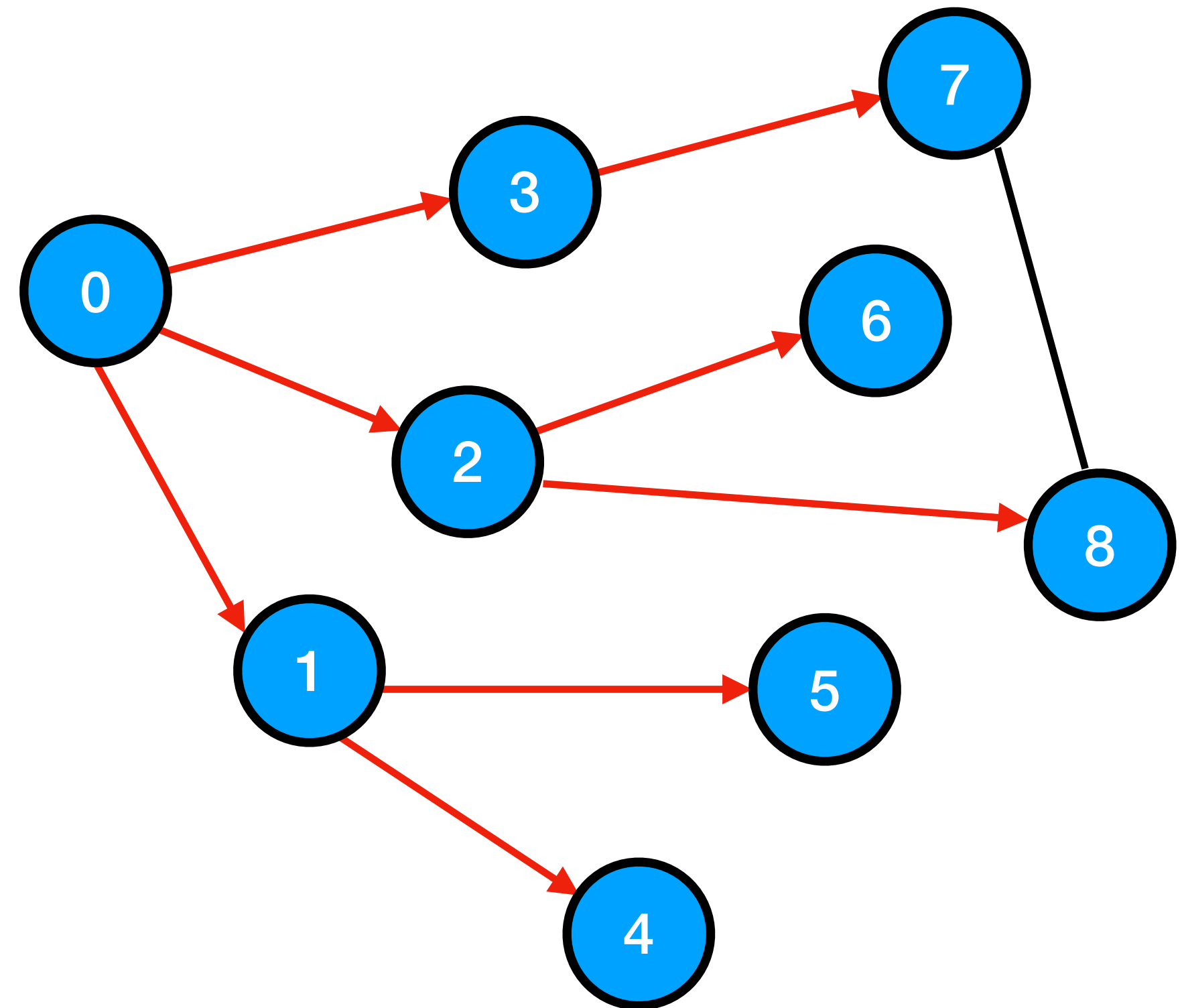
```
Adjacency list
[
  [1, 2],
  [0, 2],
  [0, 1]
]
```

# Graph Traversals

Concept and Implementation

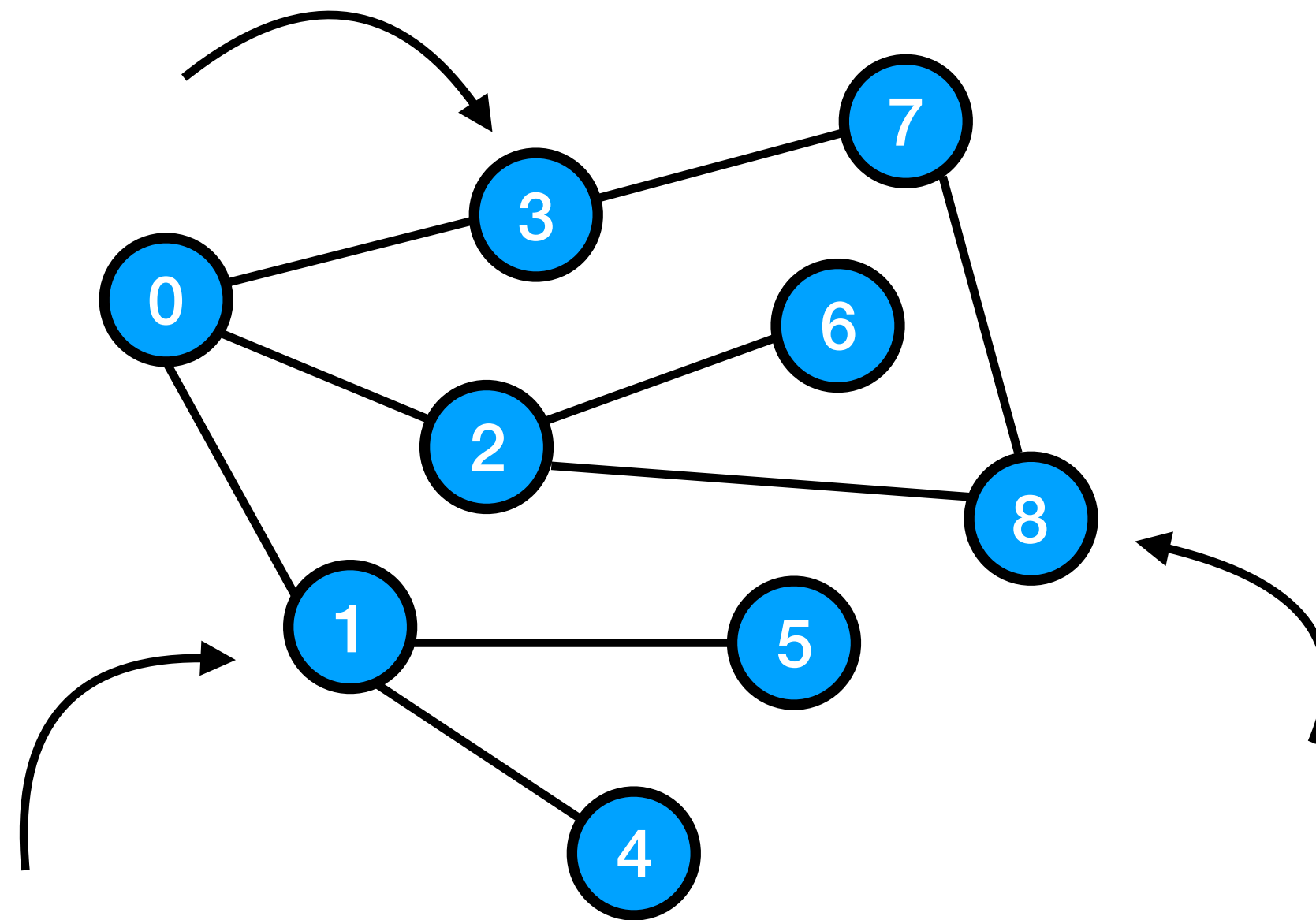
# Traverse

Process of searching through a graph by visiting vertices or nodes.



# Graph Traversal

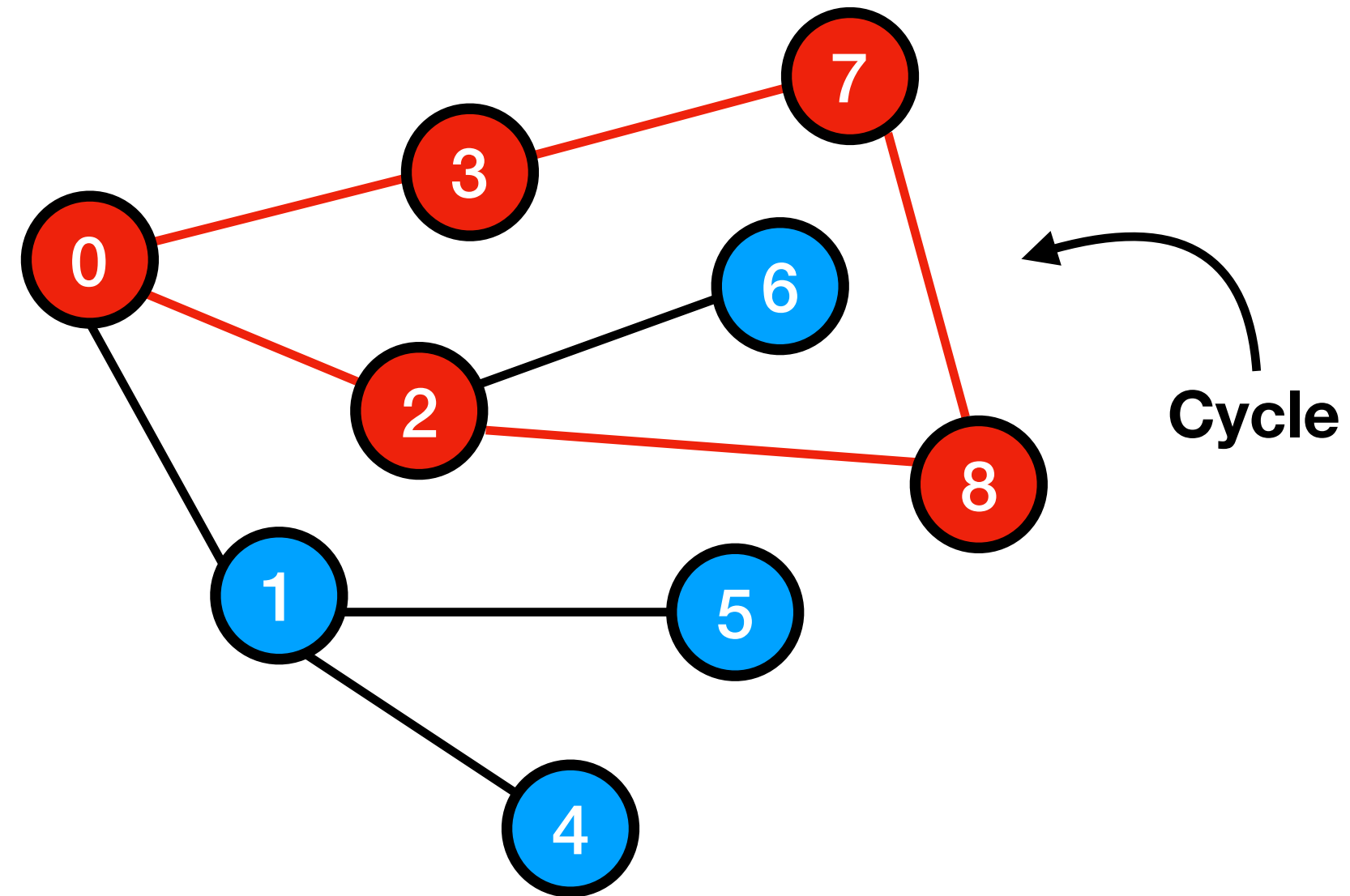
Graph traversing start with any vertex.





# Graph Traversal

Need to remember which vertex has already visited.

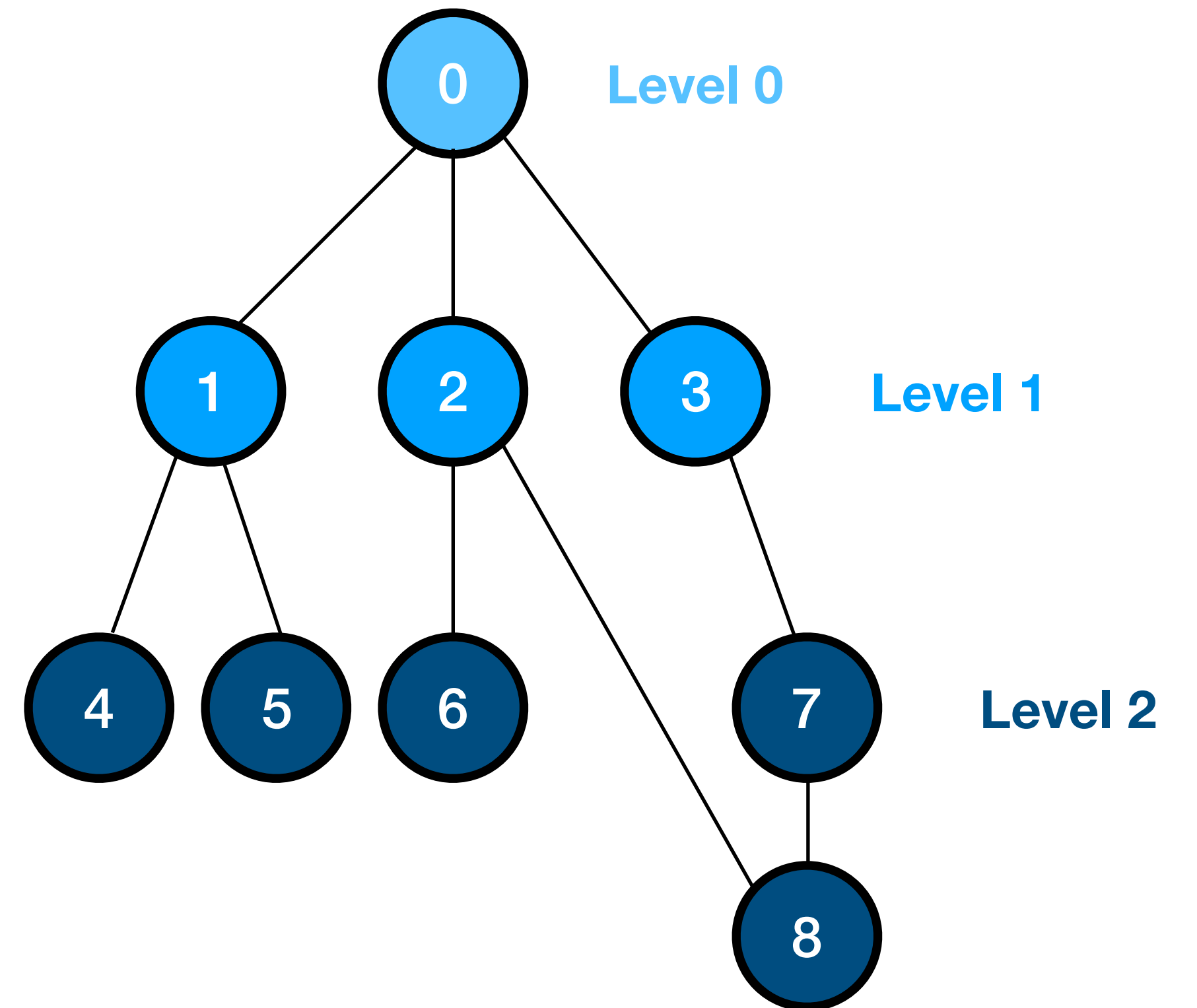


# Graph Traversal Algorithms

- Breath First Search (BFS)
- Depth First Search (DFS)

# Breath First Search (BFS)

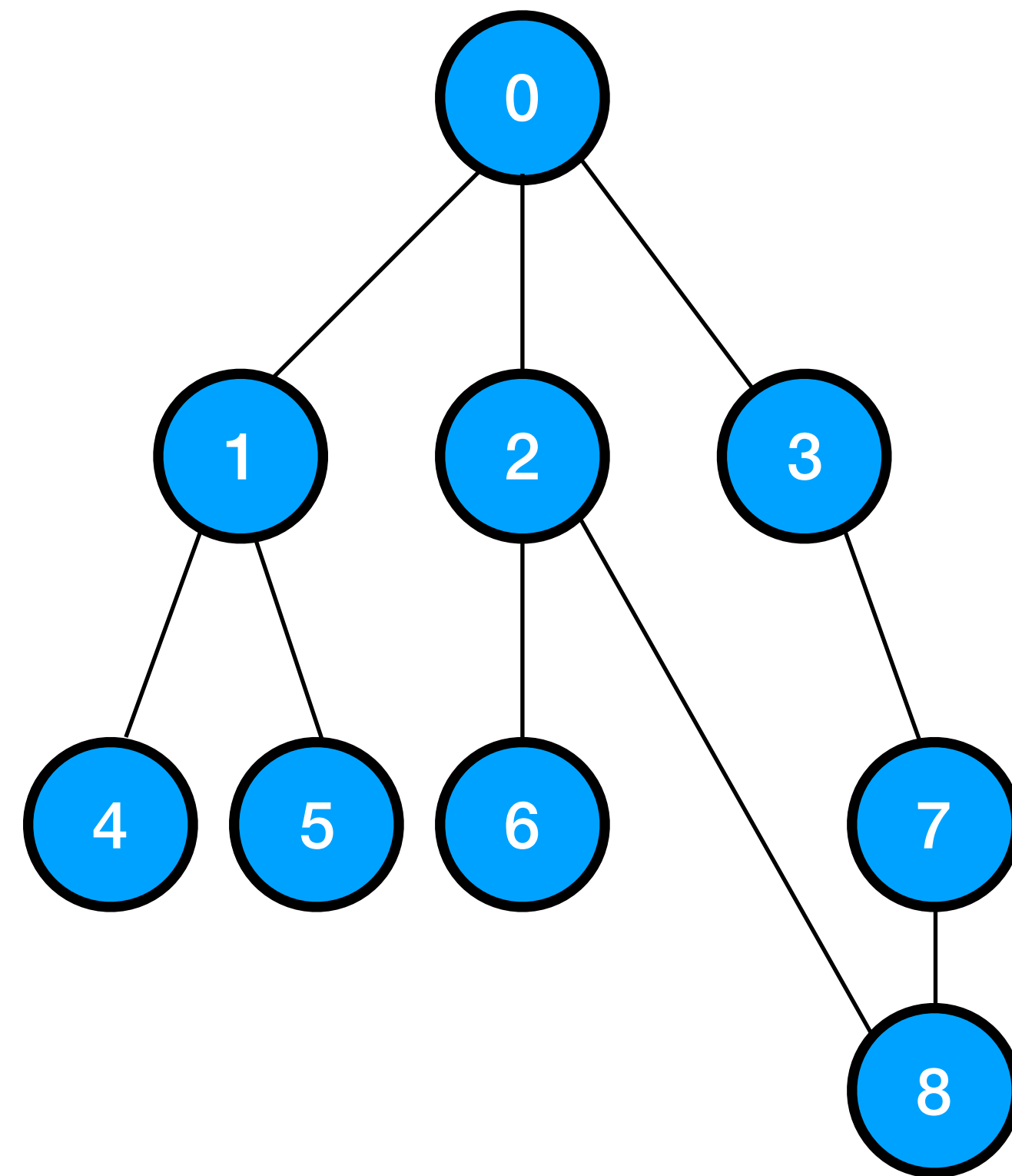
- Traverse into sibling/neighboring before children.
- Implement using queue.



# Breath First Search (BFS)



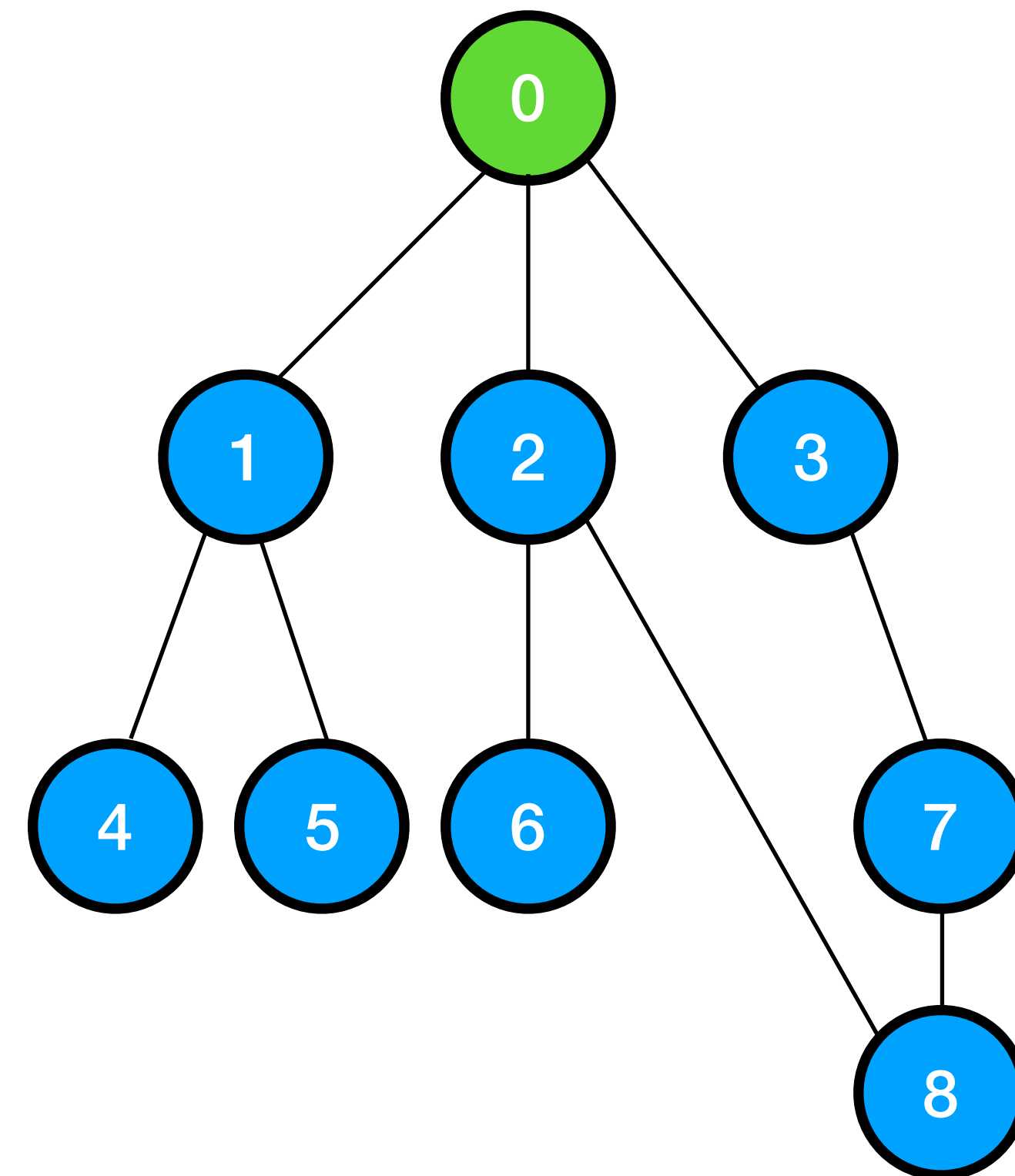
Search order :



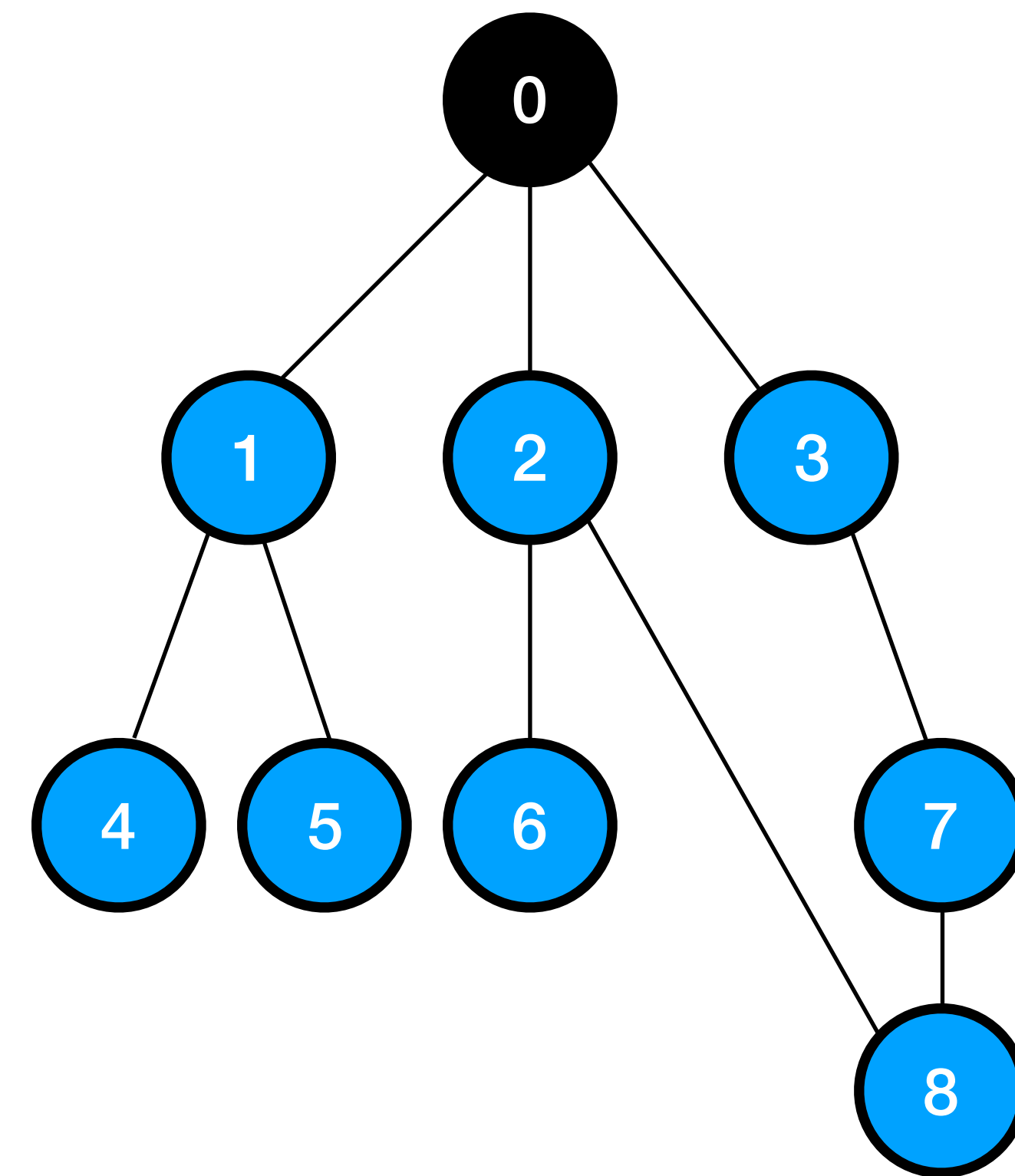
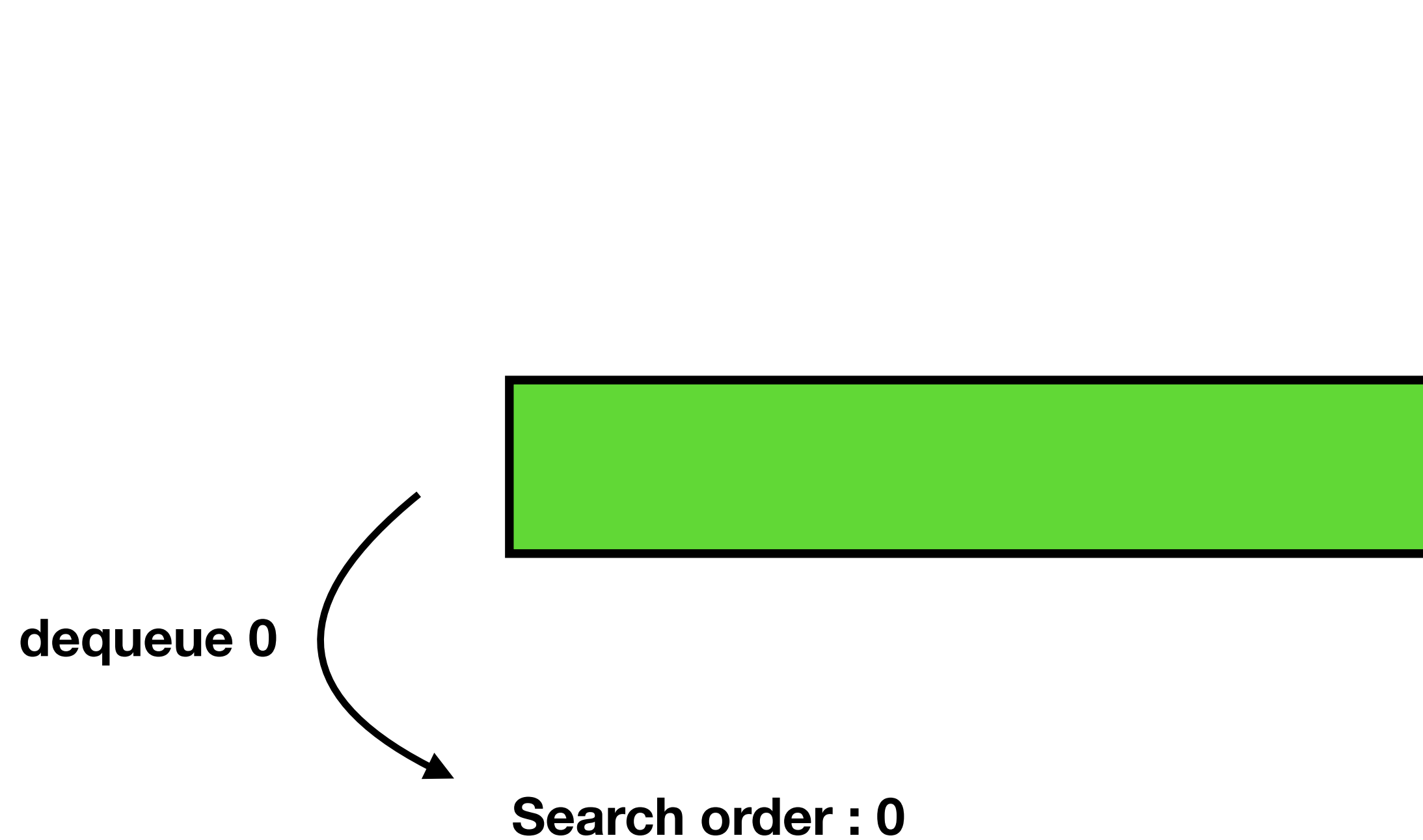
# Breath First Search (BFS)



Search order :



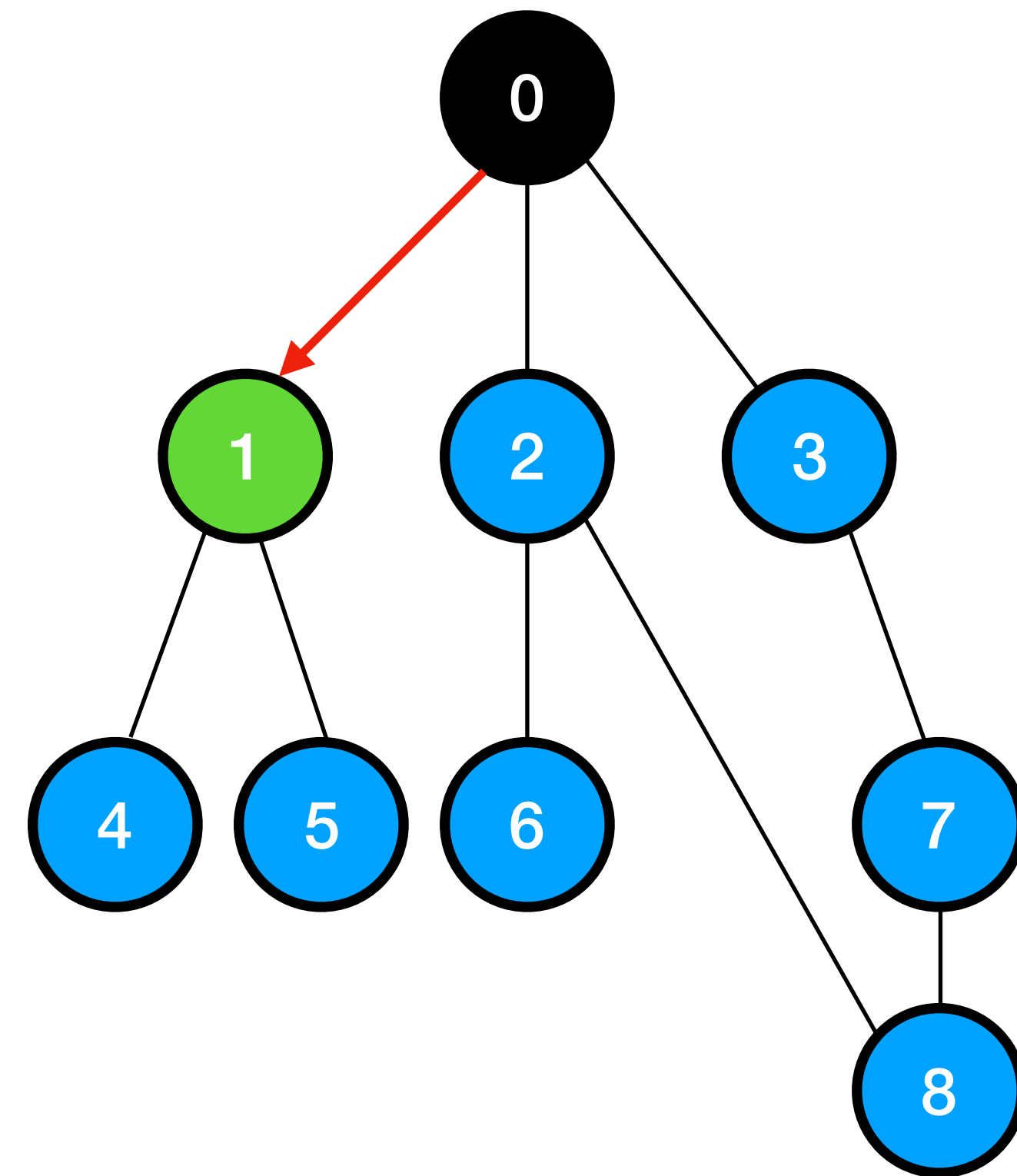
# Breath First Search (BFS)



# Breath First Search (BFS)



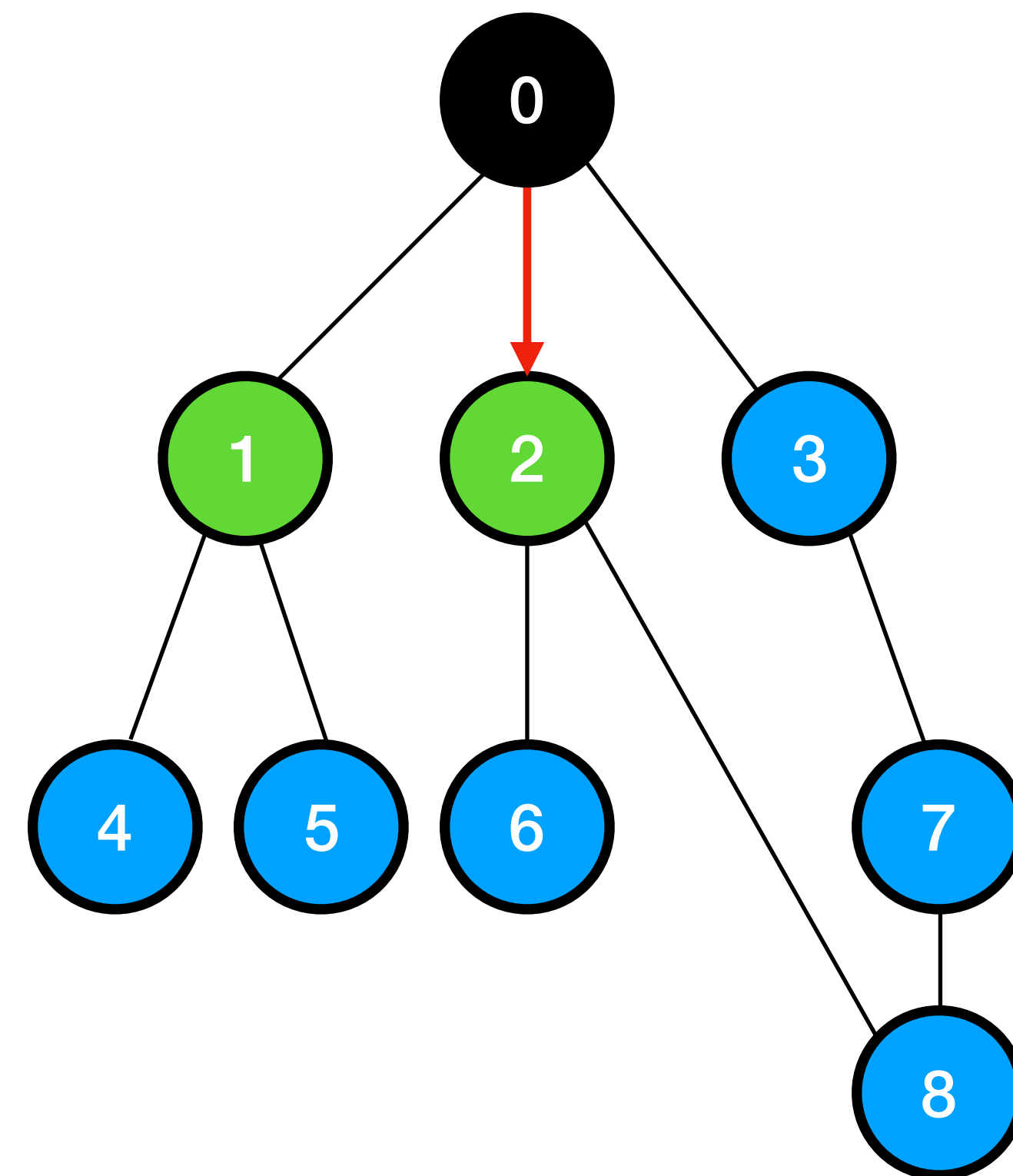
Search order : 0



# Breath First Search (BFS)



Search order : 0

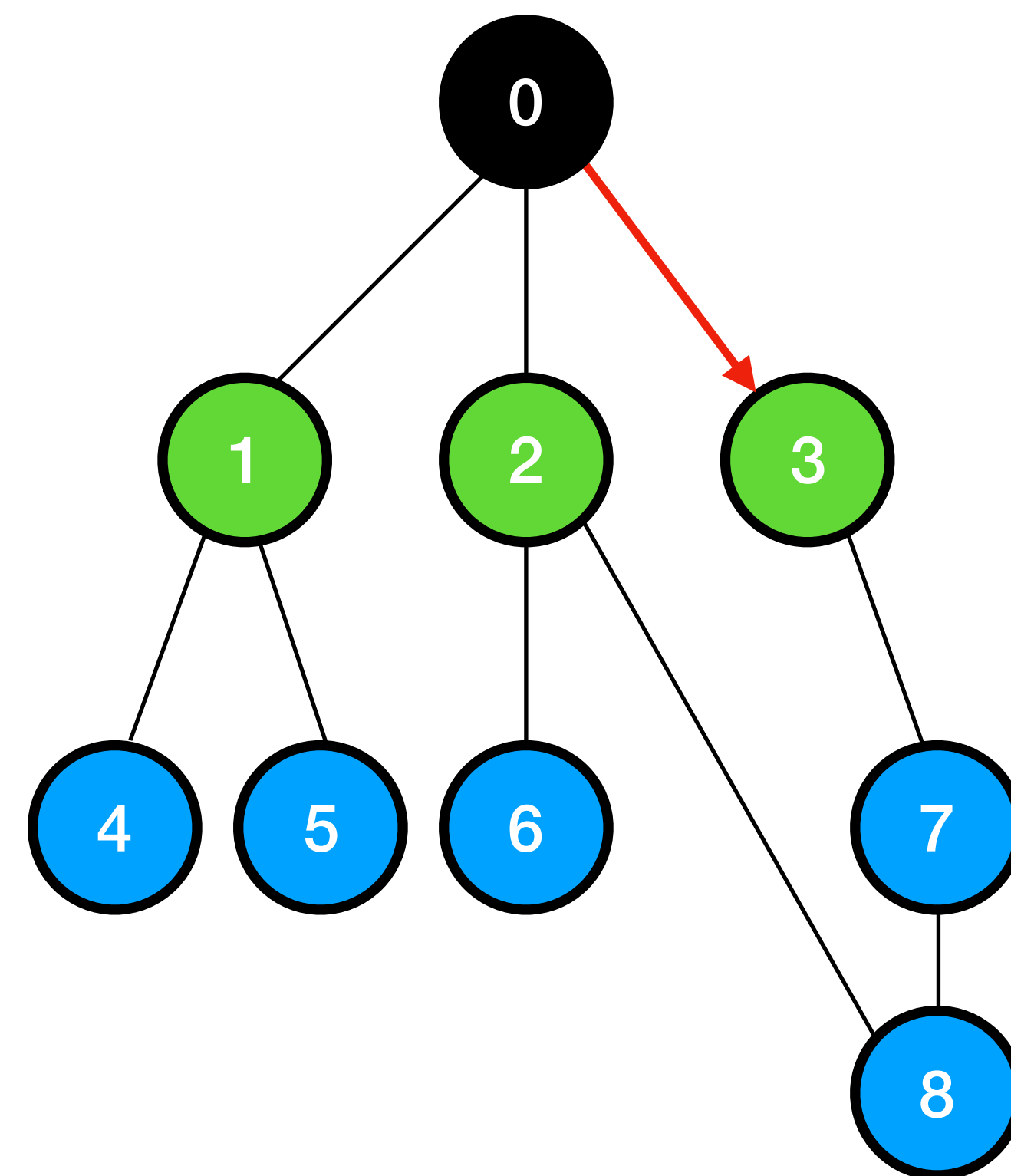




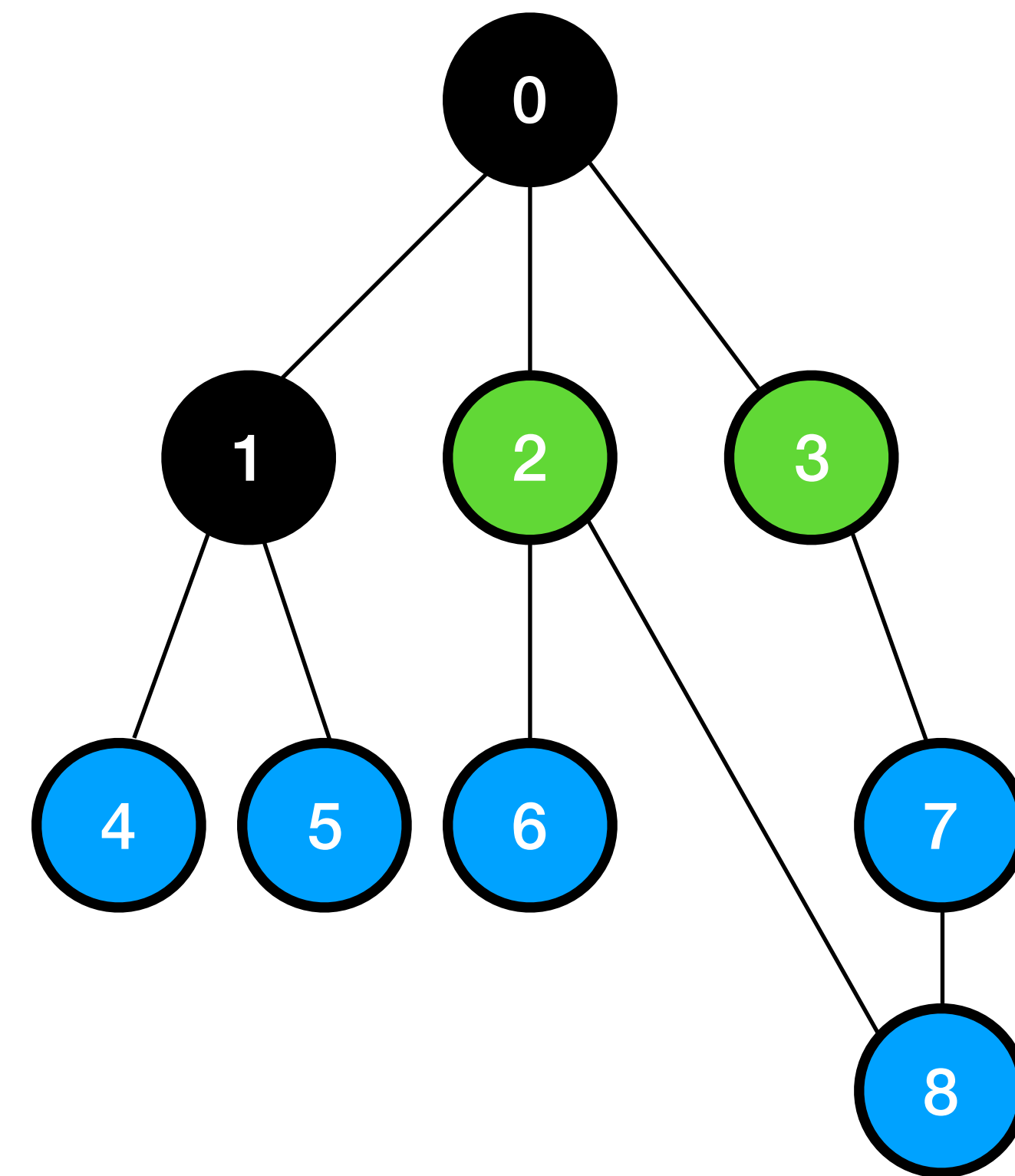
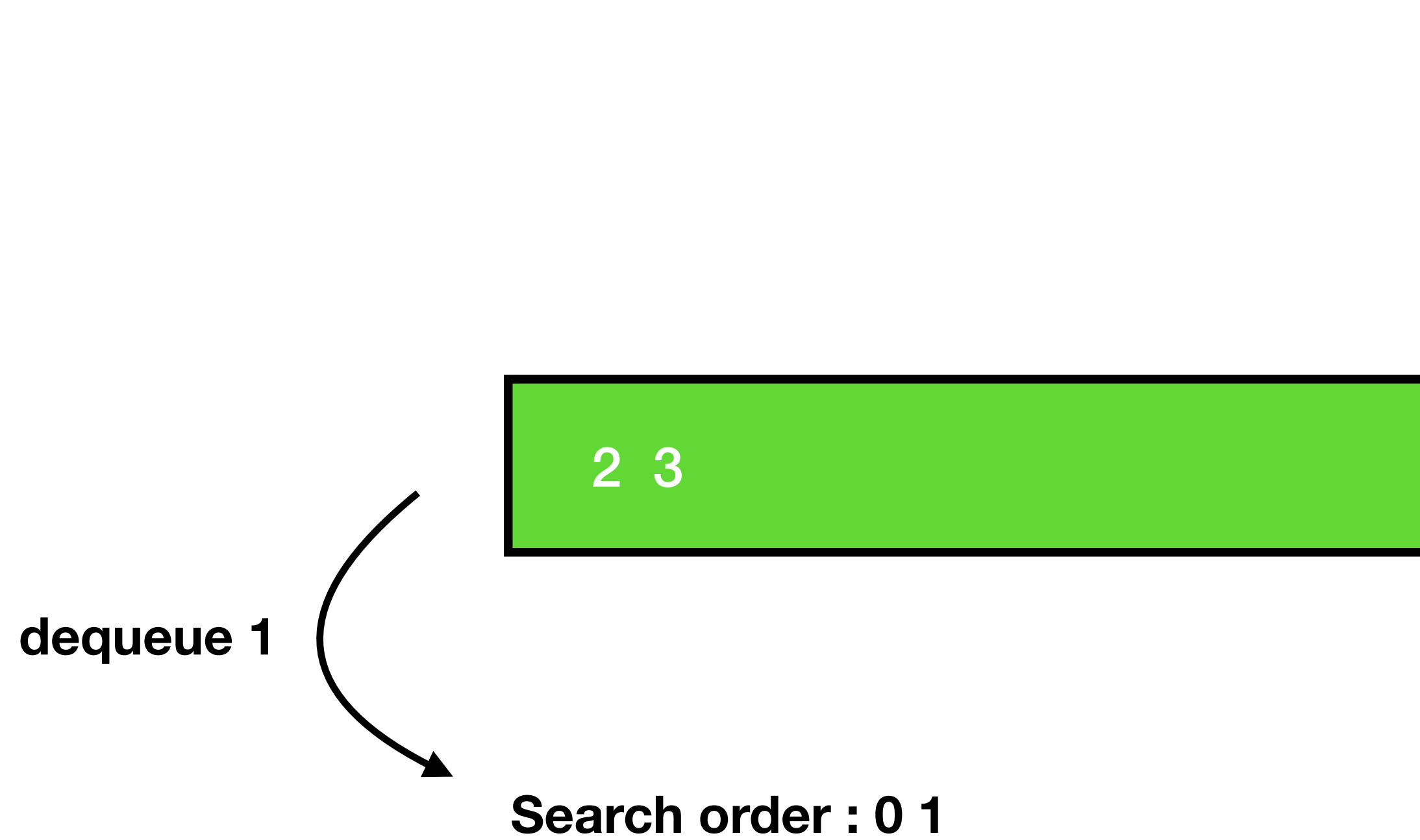
# Breath First Search (BFS)



Search order : 0



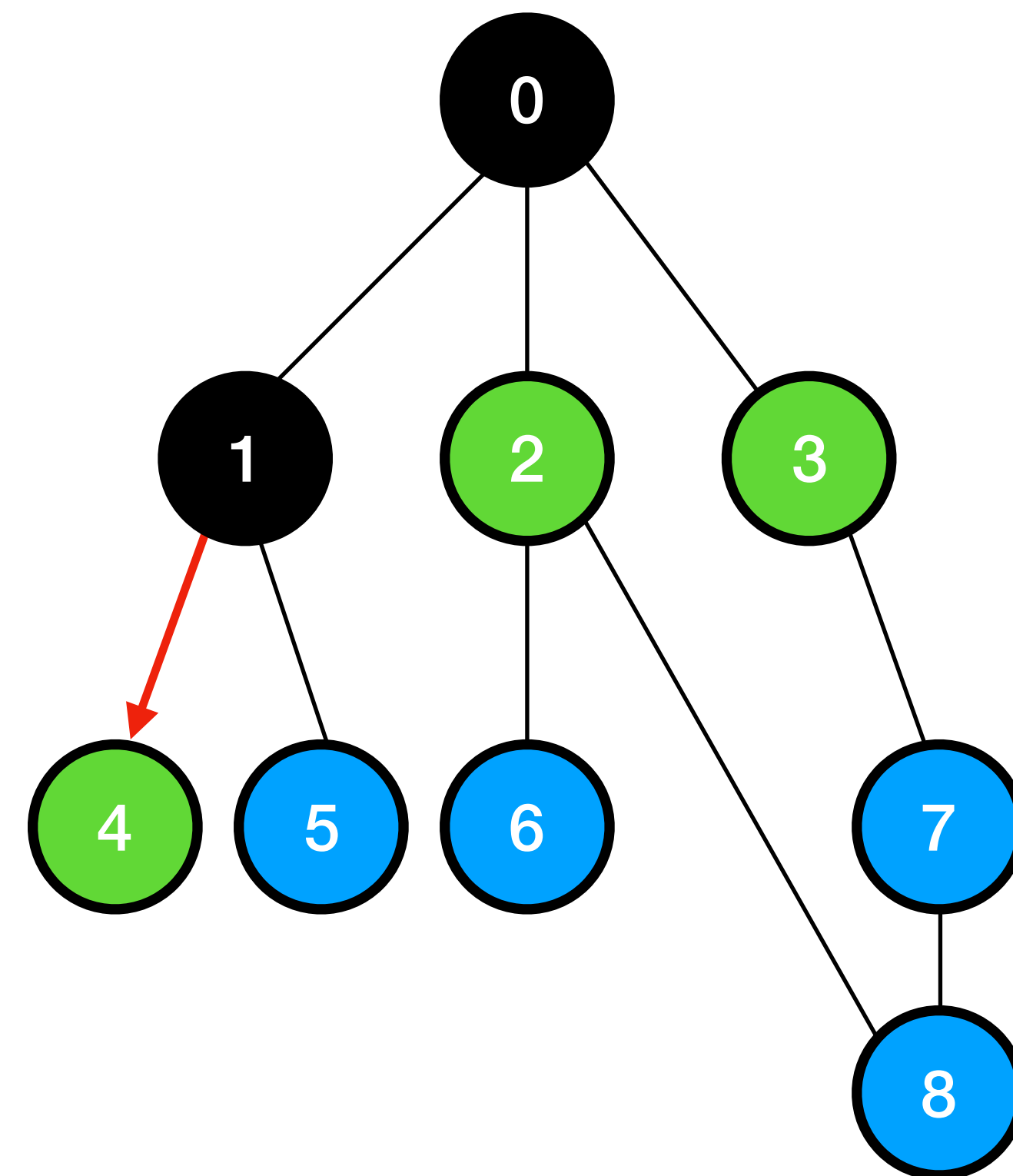
# Breath First Search (BFS)



# Breath First Search (BFS)



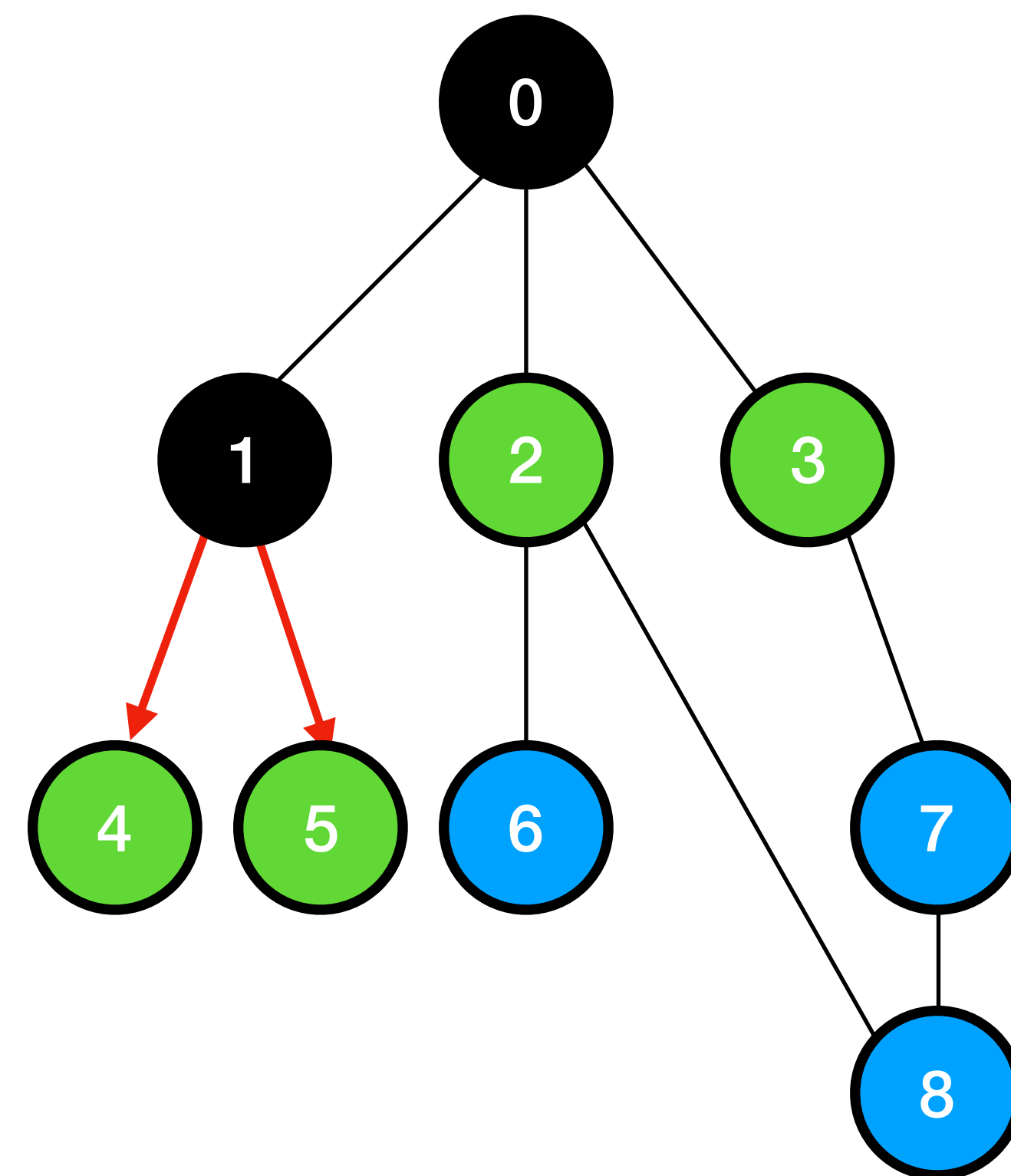
Search order : 0 1



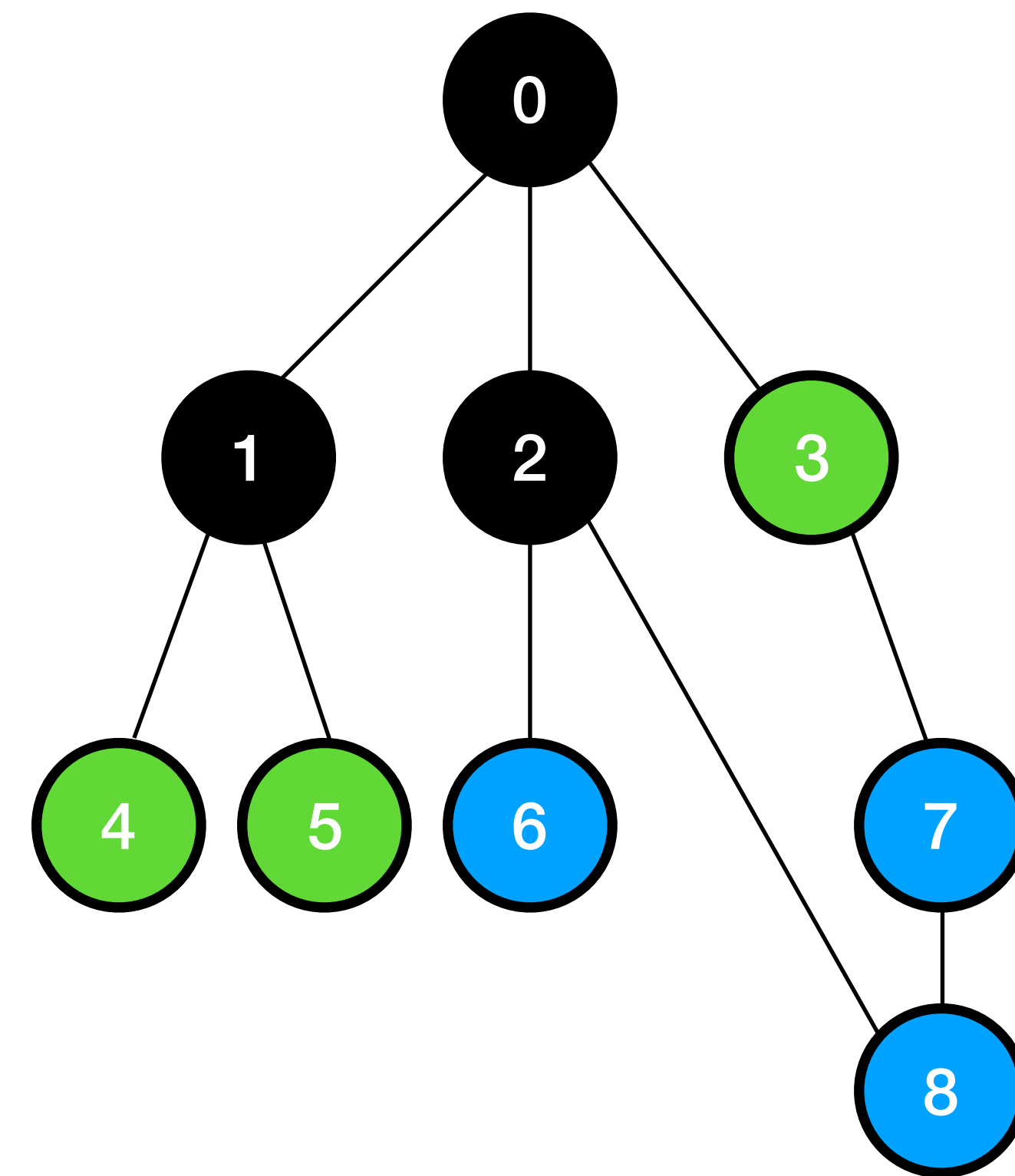
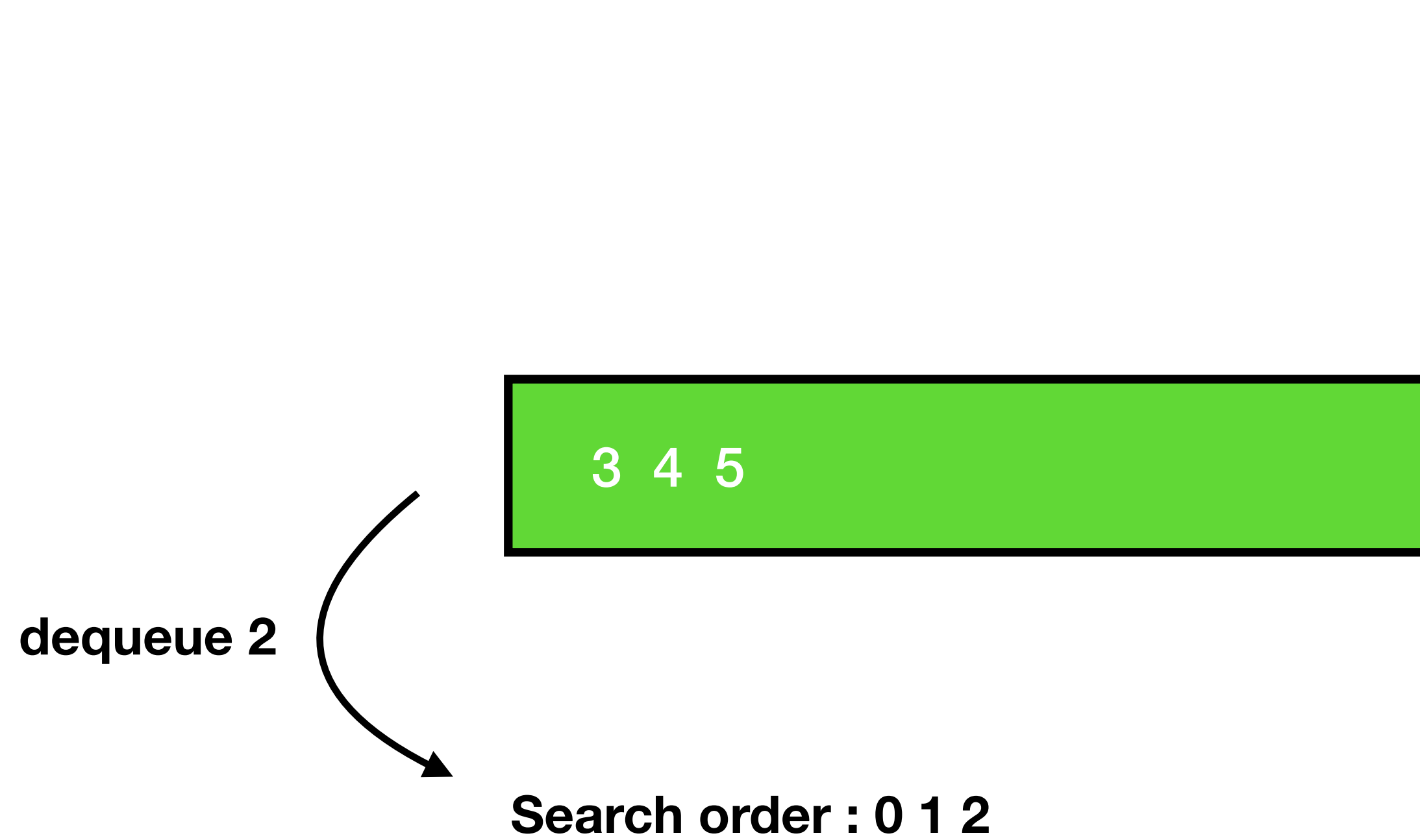
# Breath First Search (BFS)



Search order : 0 1



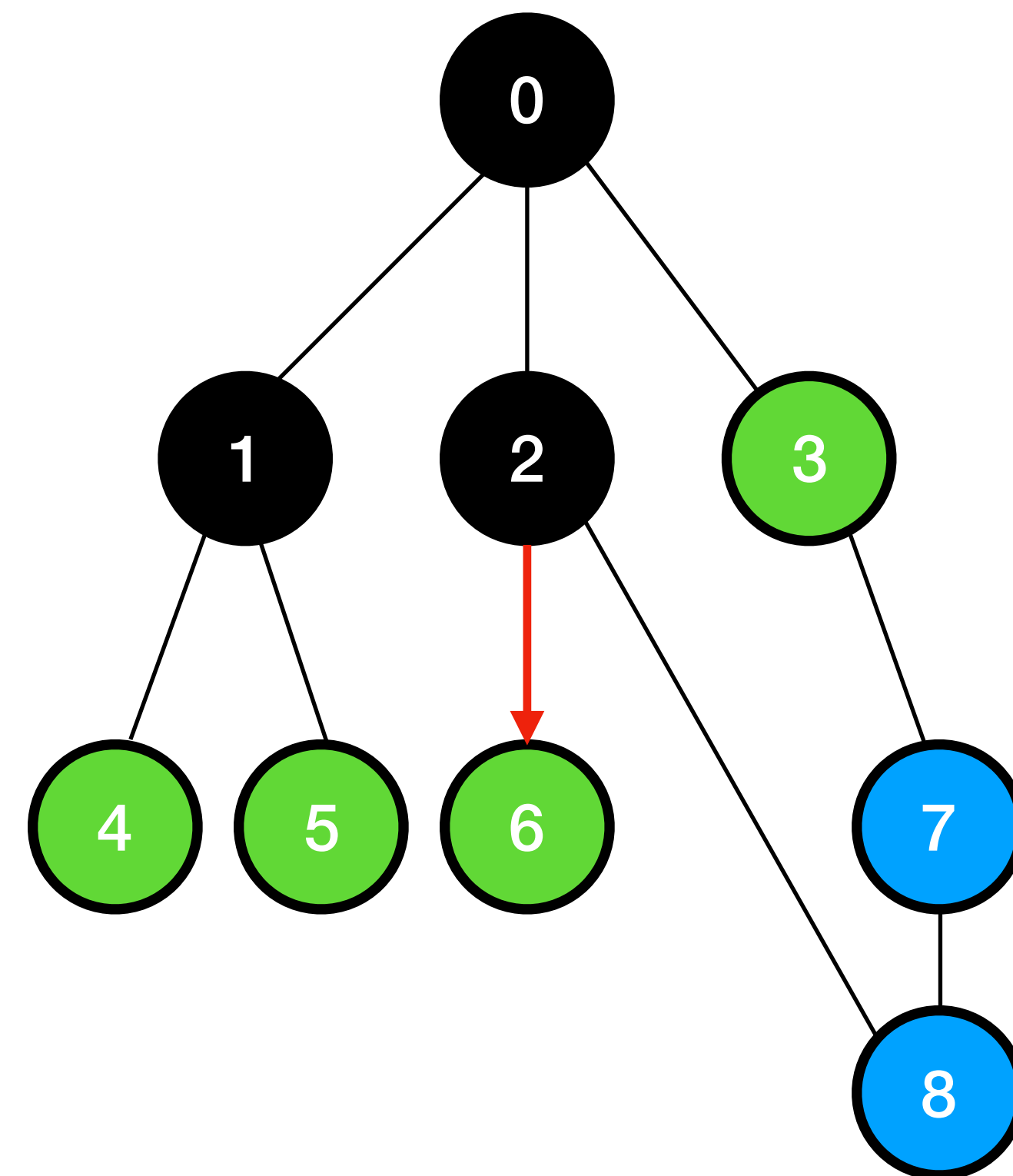
# Breath First Search (BFS)



# Breath First Search (BFS)



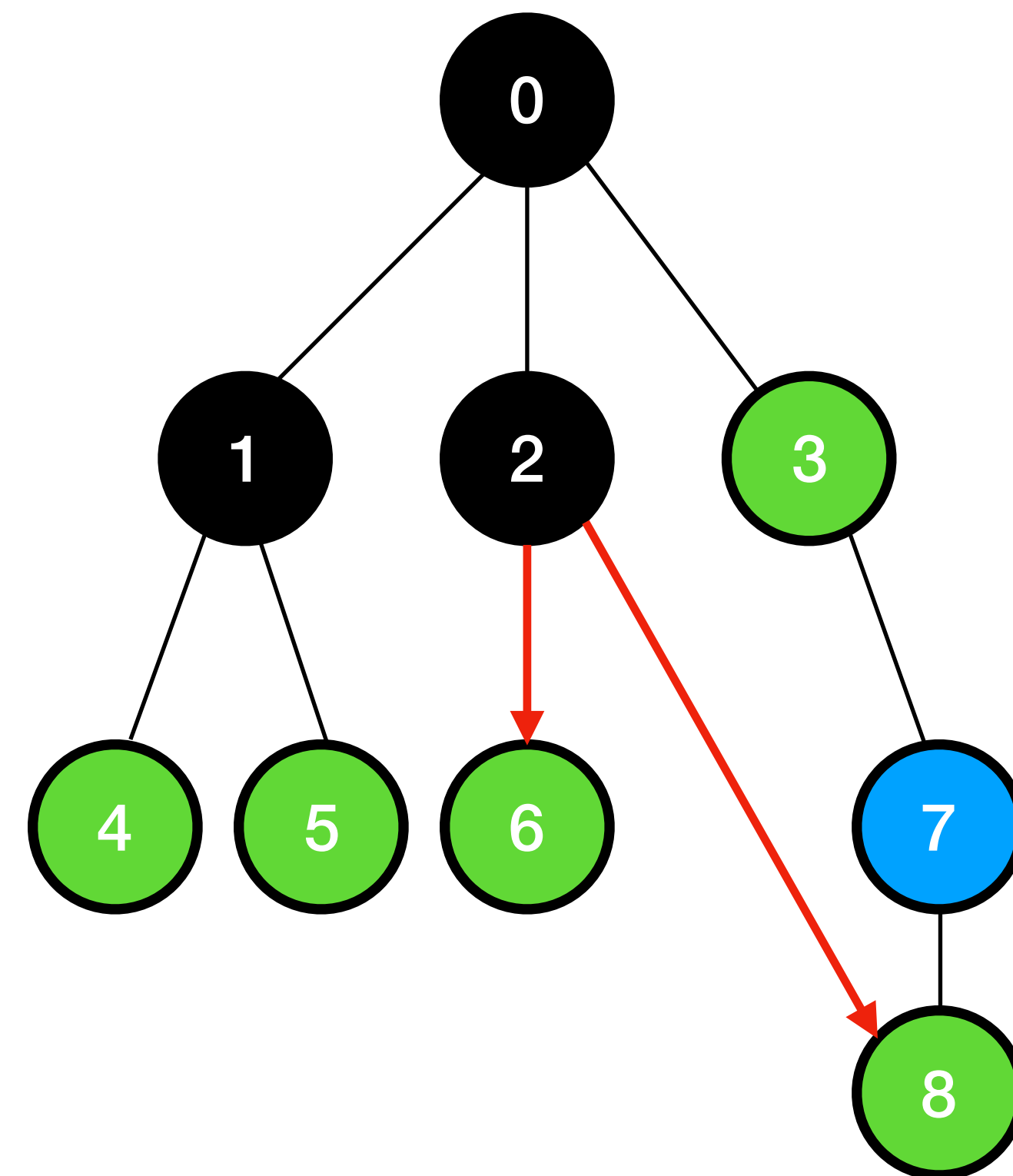
Search order : 0 1 2



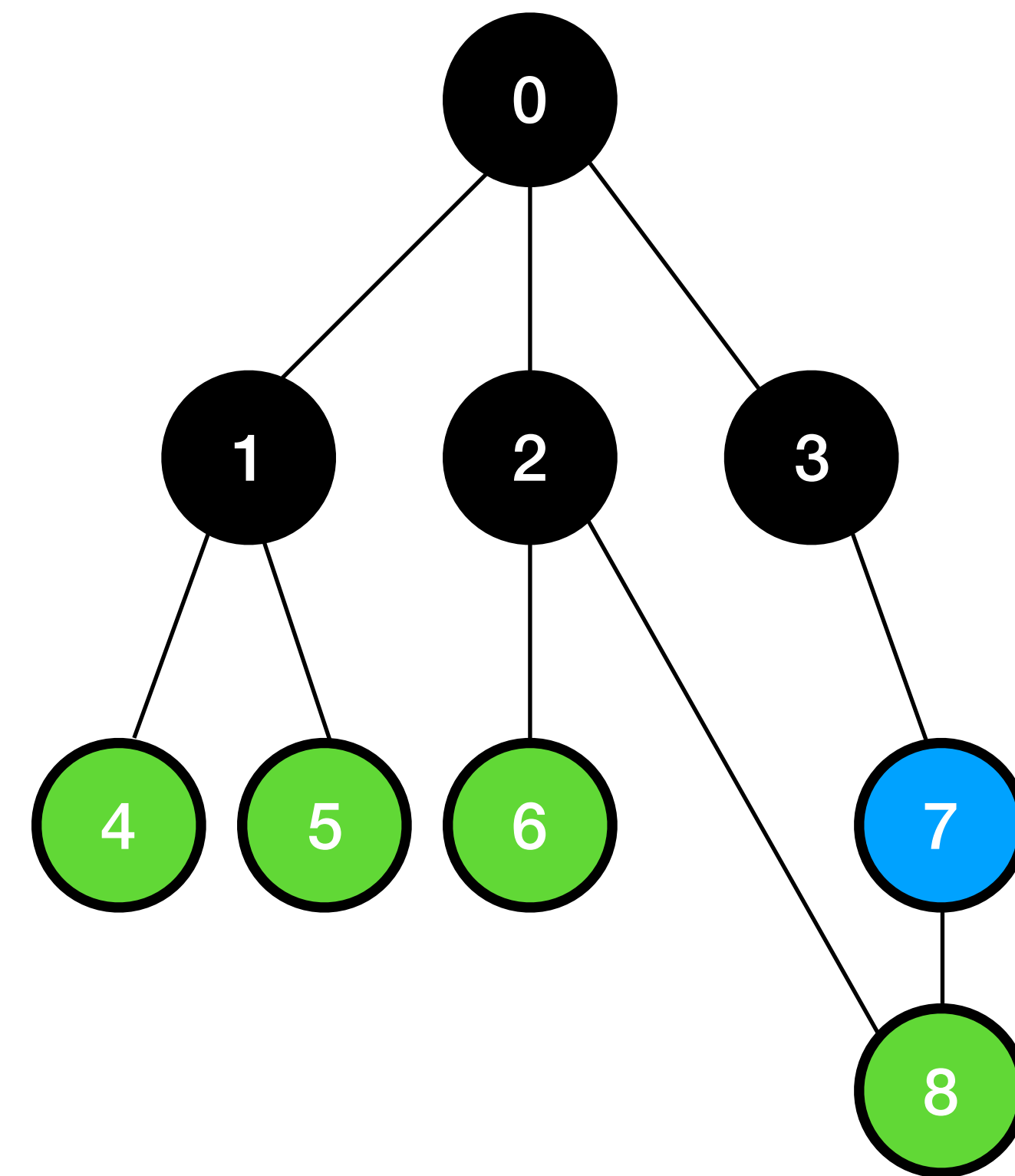
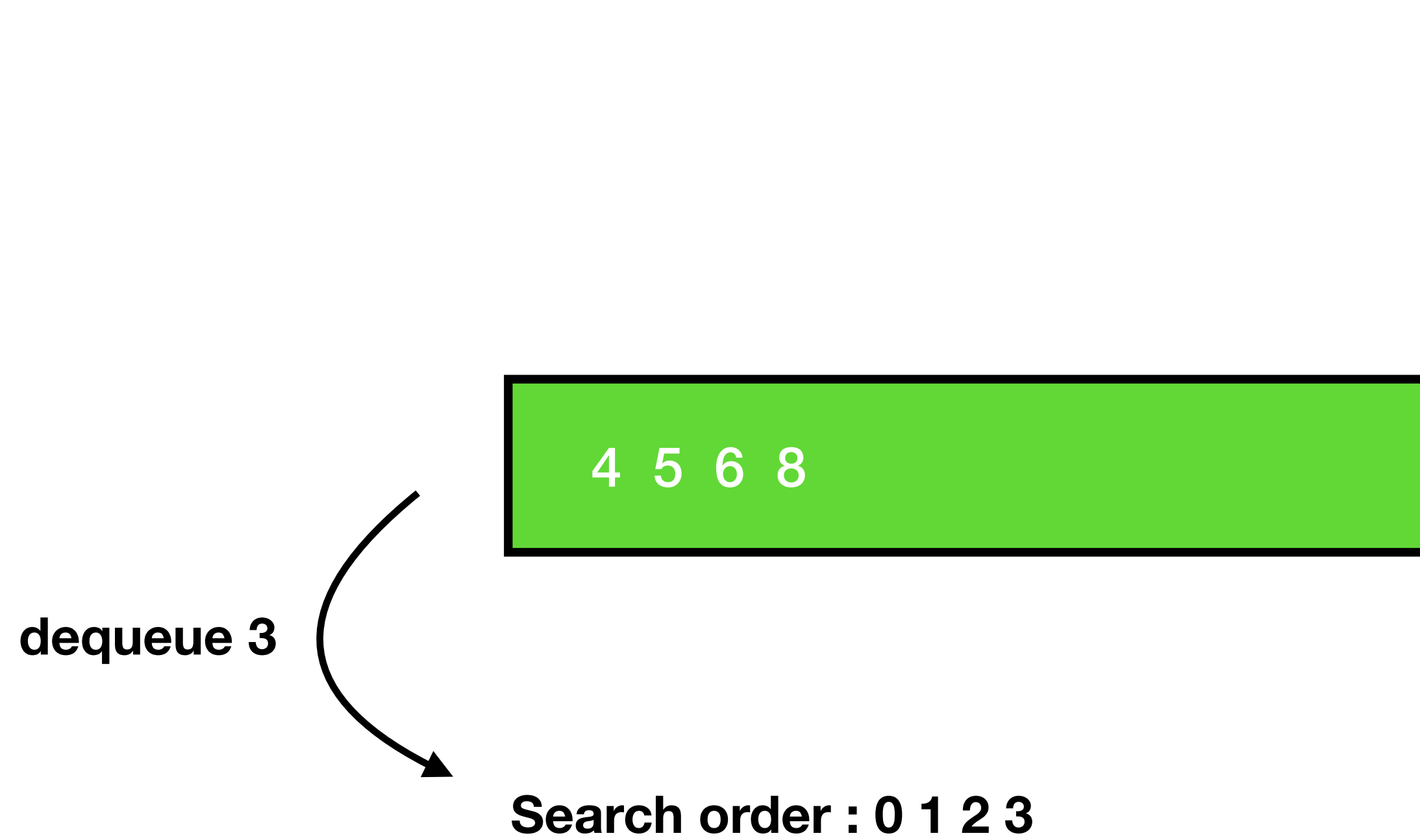
# Breath First Search (BFS)



Search order : 0 1 2



# Breath First Search (BFS)

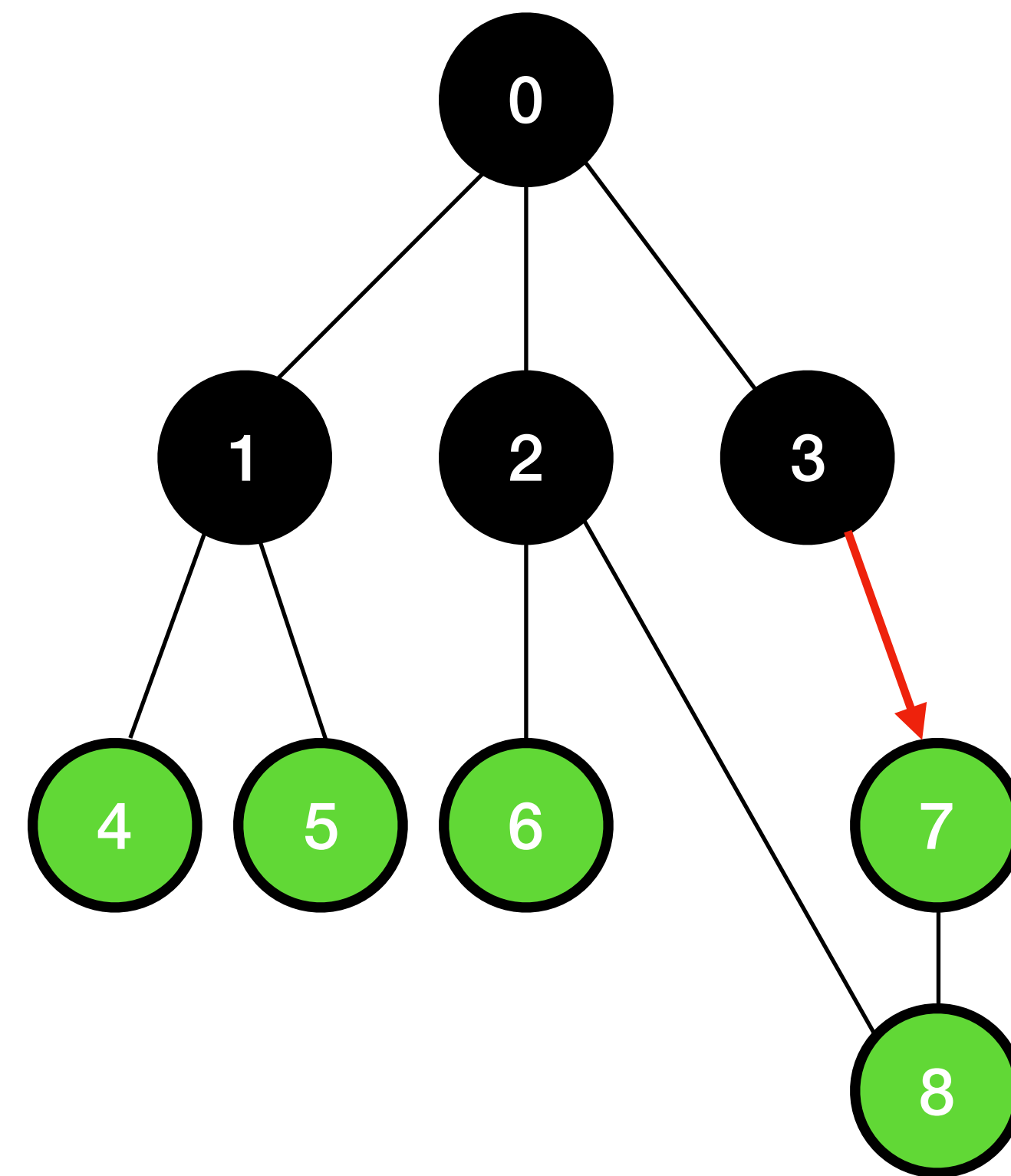




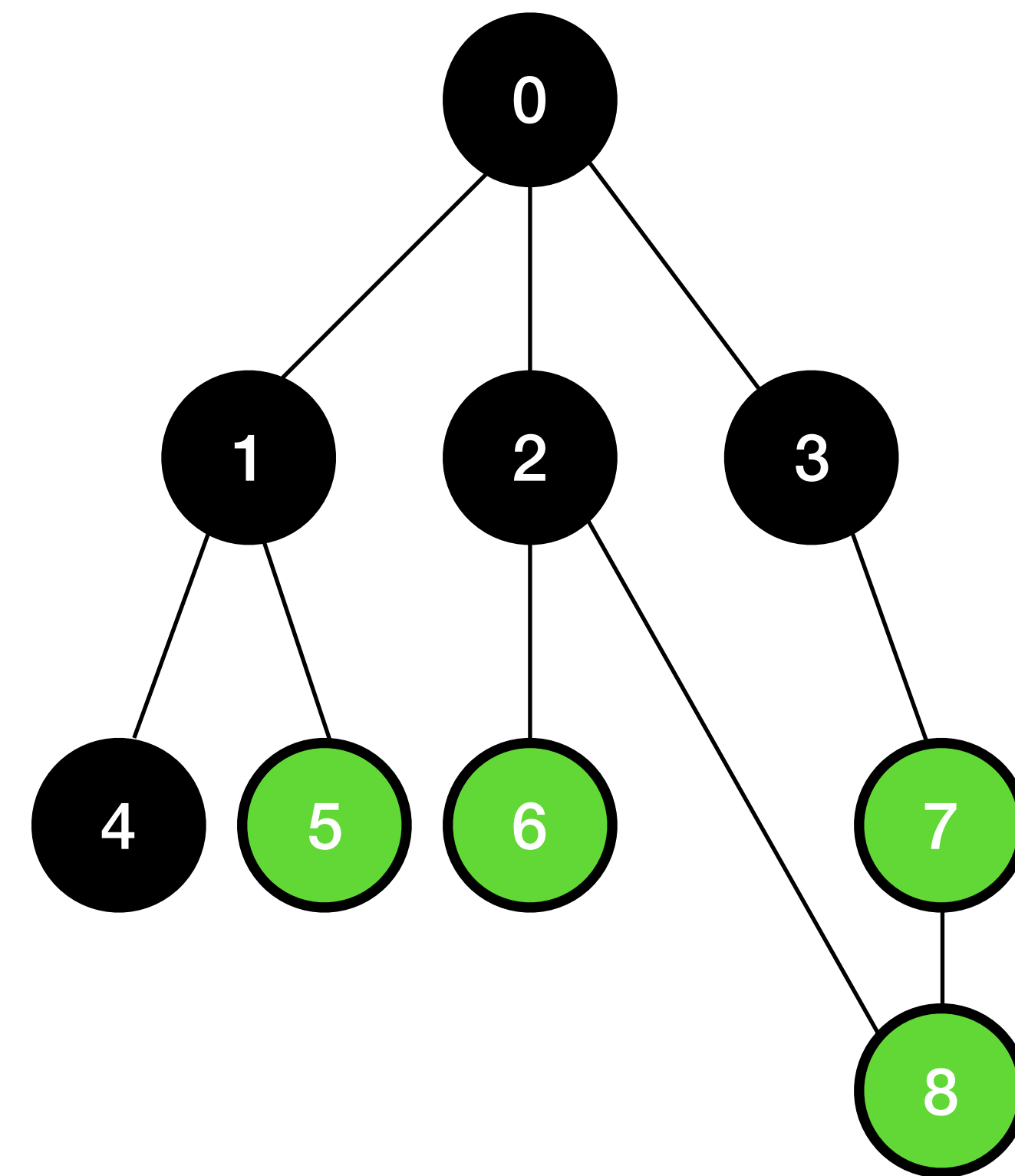
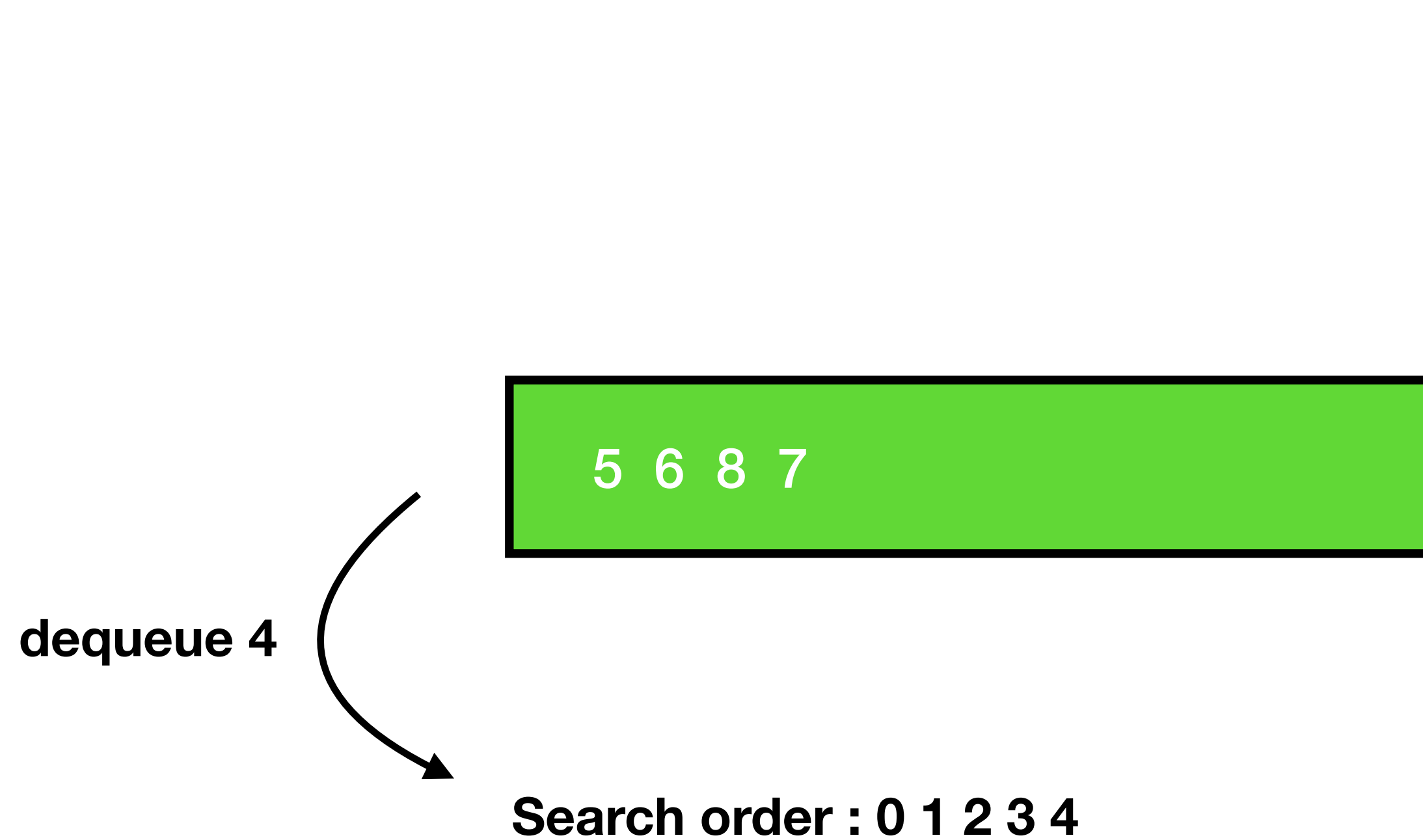
# Breath First Search (BFS)

4 5 6 8 7

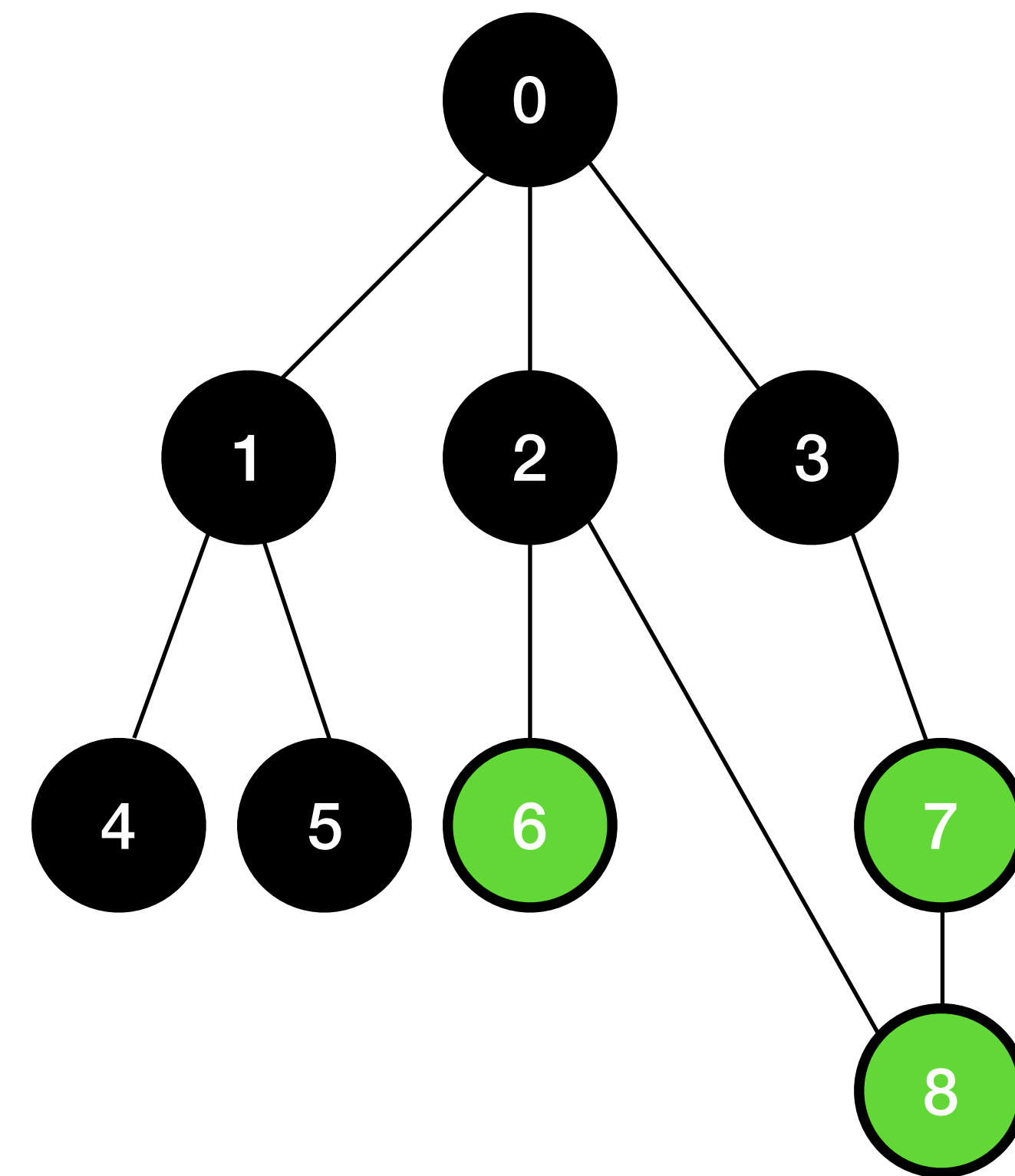
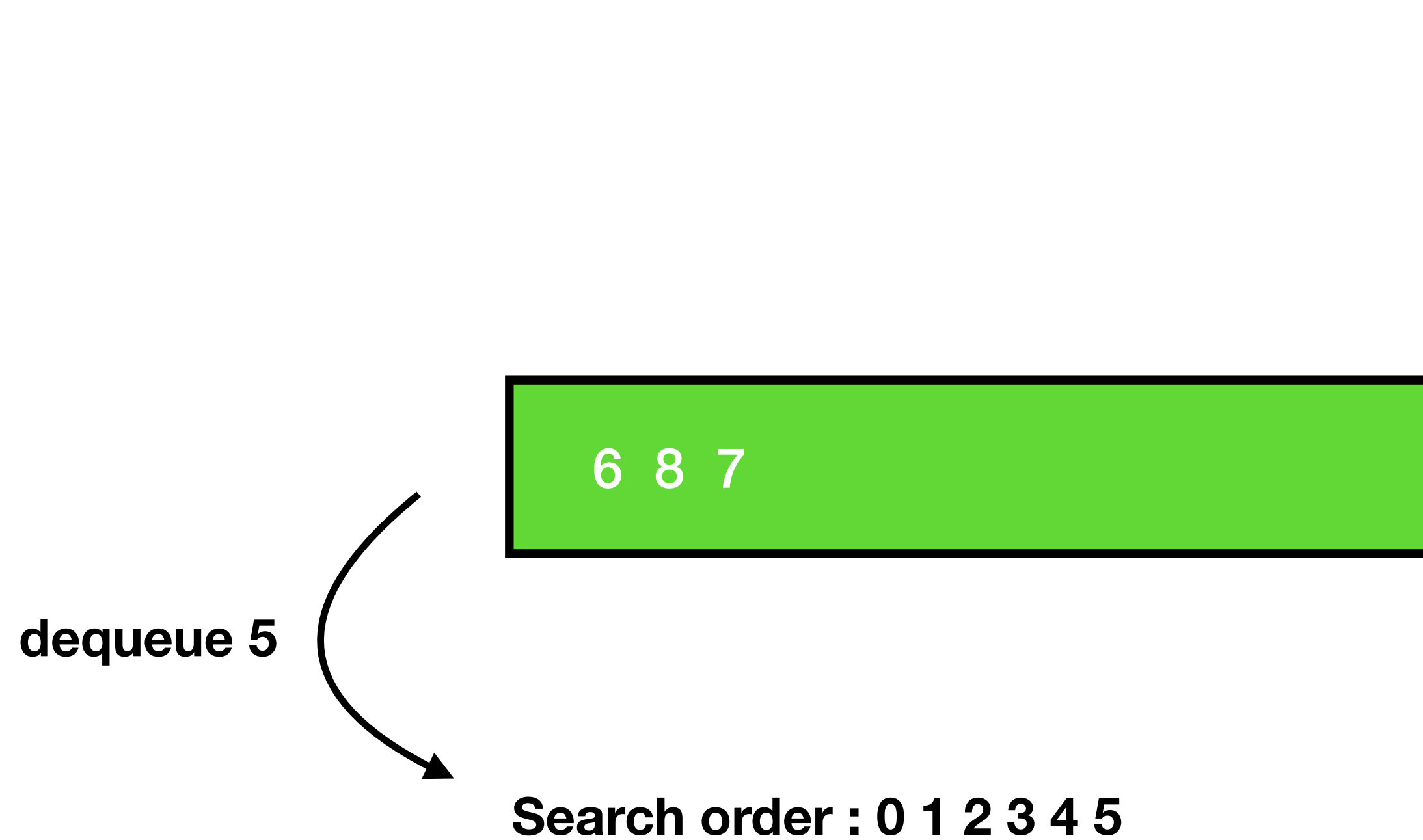
Search order : 0 1 2 3



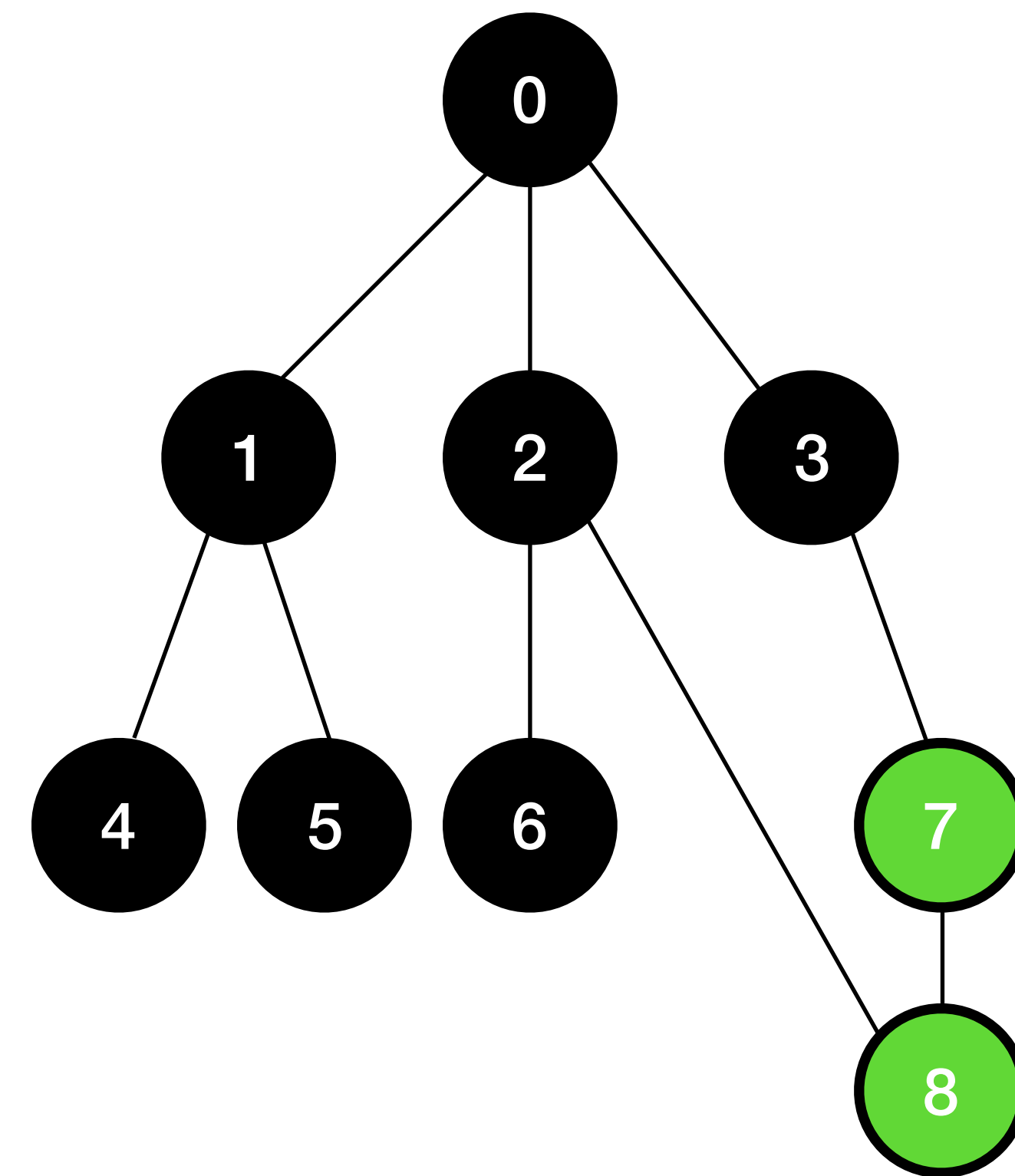
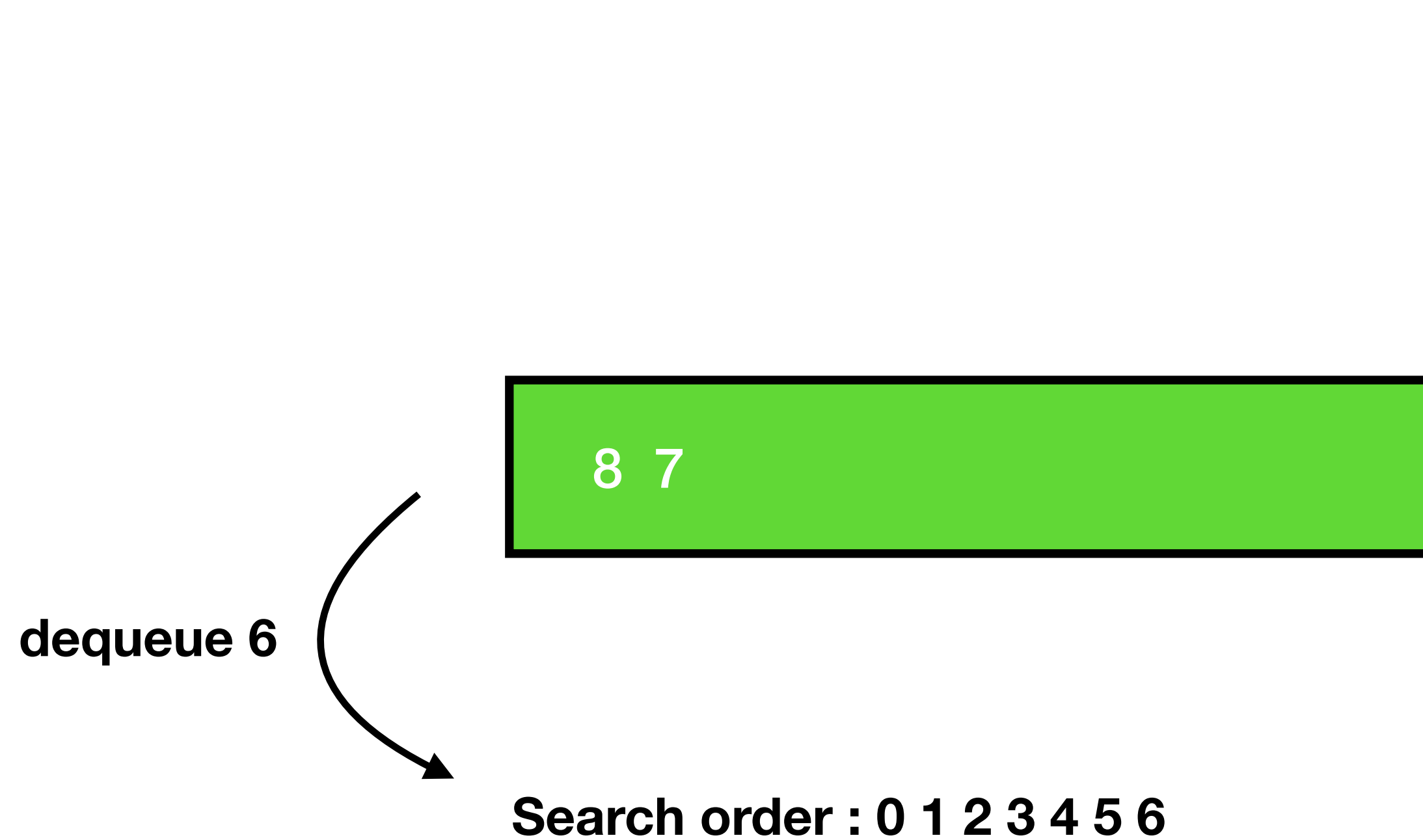
# Breath First Search (BFS)



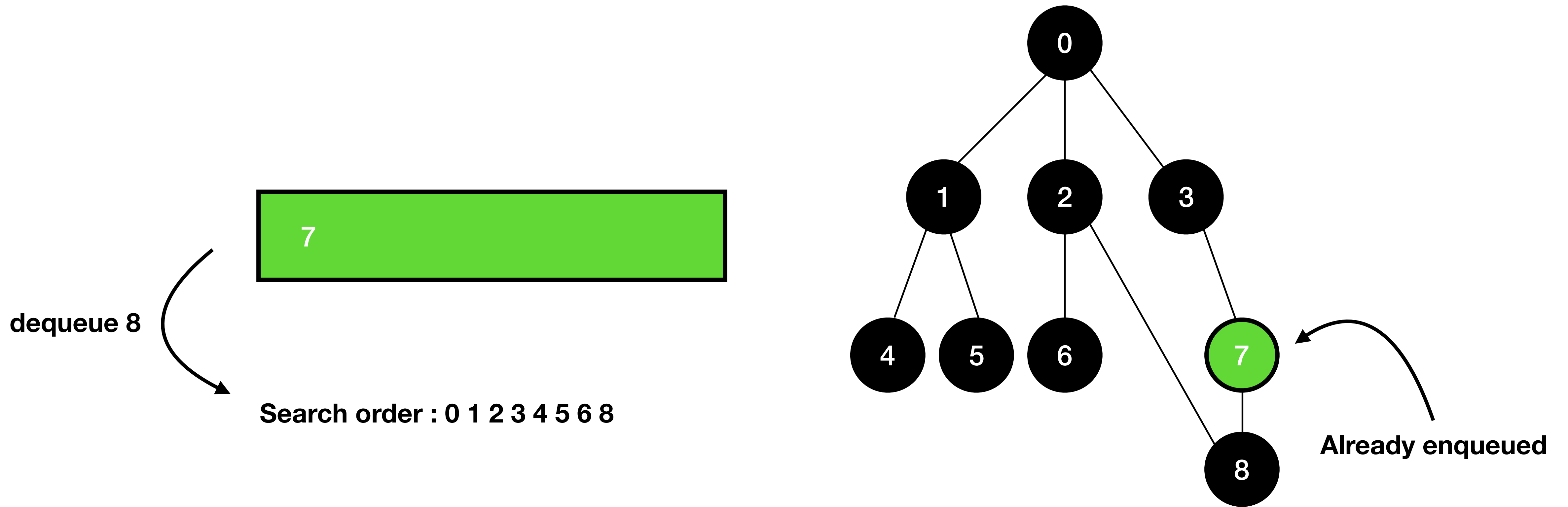
# Breath First Search (BFS)



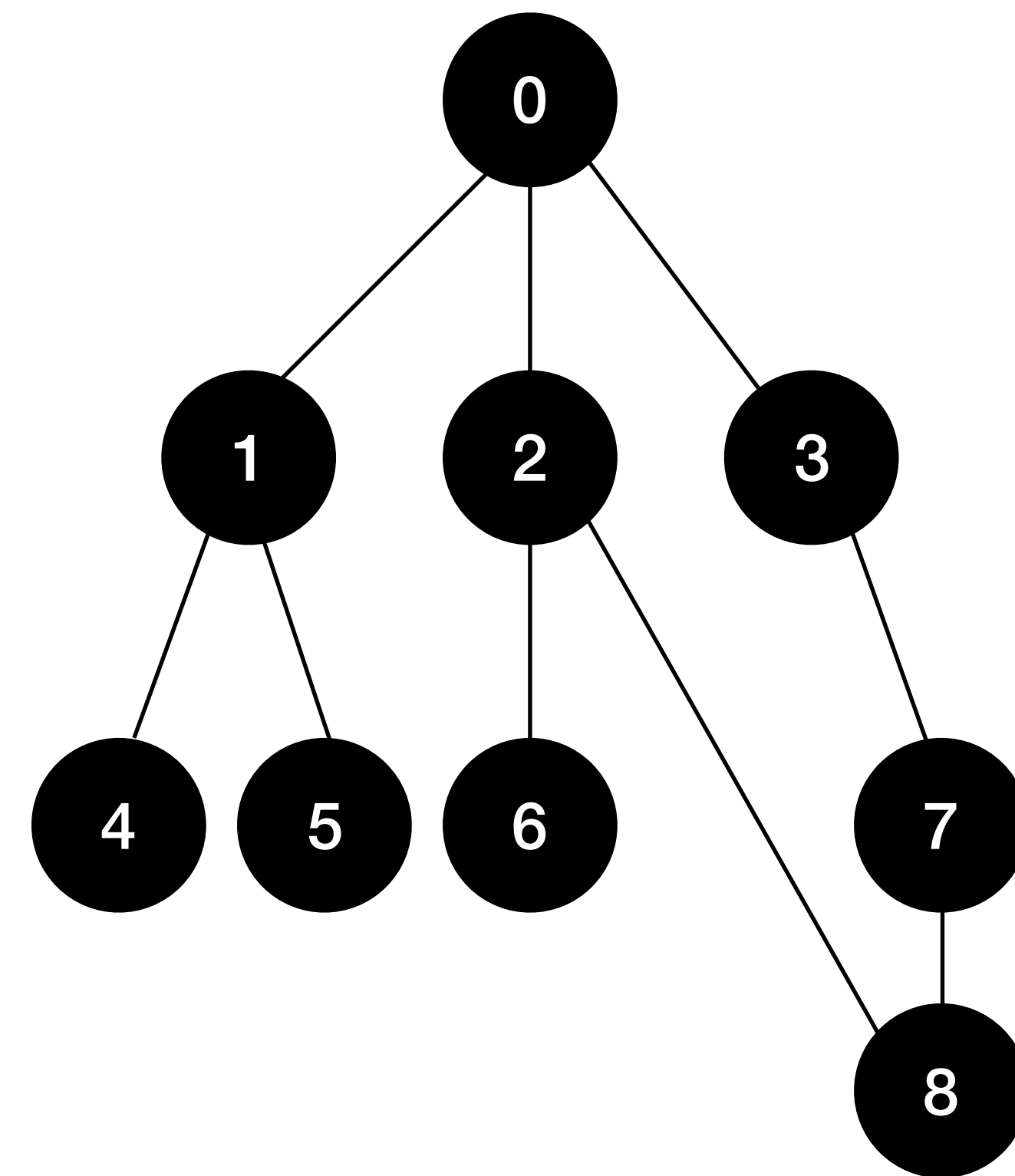
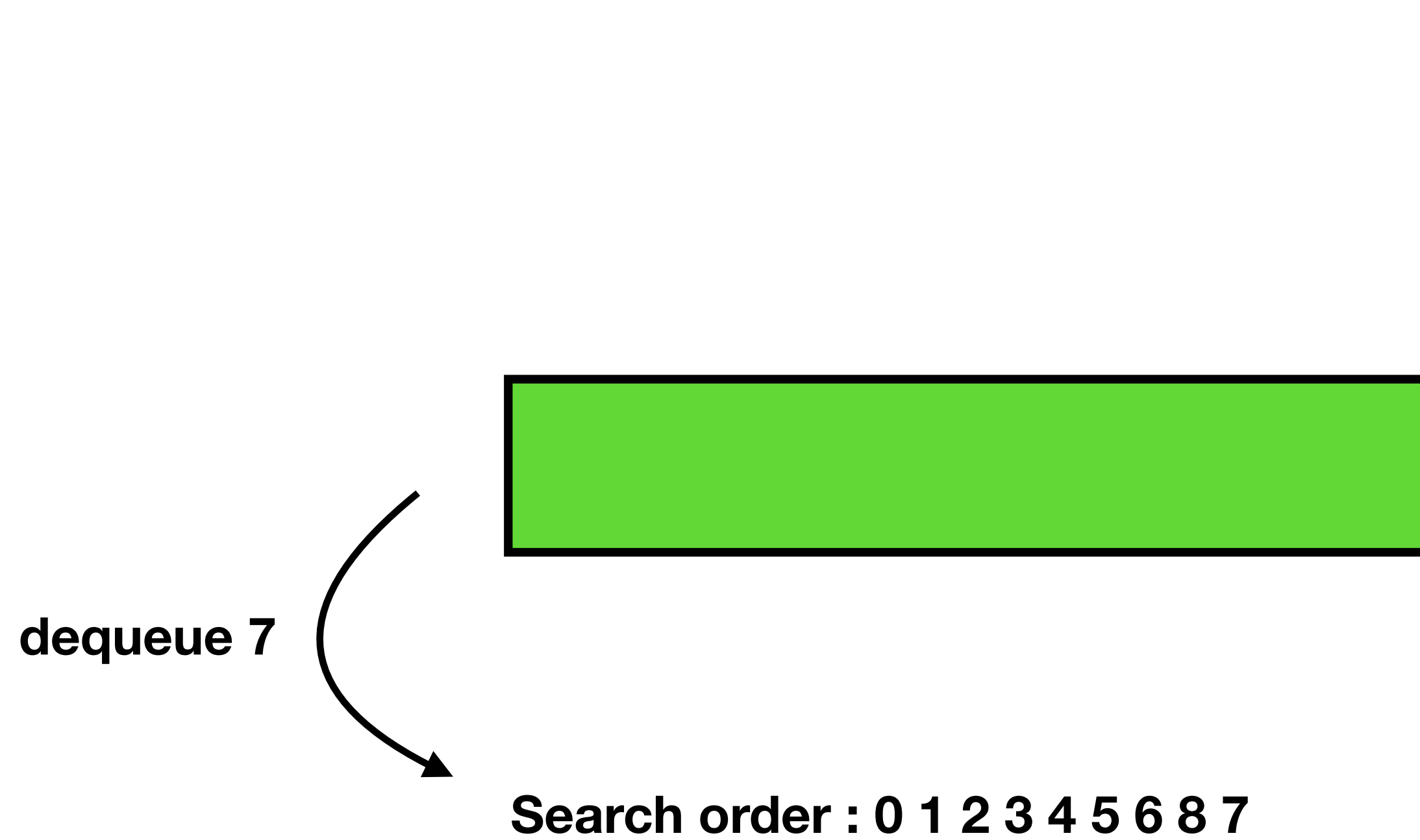
# Breath First Search (BFS)



# Breath First Search (BFS)



# Breath First Search (BFS)



# BFS Pseudocode

**bfs(startVertex)**

for vertex in vertices

    vertex.visited = false

queue = new Queue

queue.enqueue(startVertex)

startVertex.visited = true

while queue.isNotEmpty

    currentVertex = queue.dequeue()

    for children of currentVertex

        if children.visited == false

            queue.enqueue(children)

            children.visited = true

# BFS Pseudocode

```
bfs(startVertex)
```

```
  for vertex in vertices
```

```
    vertex.visited = false
```

```
  queue = new Queue
```

```
  queue.enqueue(startVertex)
```

```
  startVertex.visited = true
```

```
  while queue.isNotEmpty
```

```
    currentVertex = queue.dequeue()
```

```
    for children of currentVertex
```

```
      if children.visited == false
```

```
        queue.enqueue(children)
```

```
        children.visited = true
```



# BFS Pseudocode

```
bfs(startVertex)
  for vertex in vertices
    vertex.visited = false
  queue = new Queue
  queue.enqueue(startVertex)
  startVertex.visited = true

  while queue.isNotEmpty
    currentVertex = queue.dequeue()
    for children of currentVertex
      if children.visited == false
        queue.enqueue(children)
        children.visited = true
```

# BFS Pseudocode

```
bfs(startVertex)
  for vertex in vertices
    vertex.visited = false
  queue = new Queue
  queue.enqueue(startVertex)
  startVertex.visited = true

  while queue.isNotEmpty
    currentVertex = queue.dequeue()
    for children of currentVertex
      if children.visited == false
        queue.enqueue(children)
        children.visited = true
```

# BFS Pseudocode

```
bfs(startVertex)
  for vertex in vertices
    vertex.visited = false
  queue = new Queue
  queue.enqueue(startVertex)
  startVertex.visited = true

  while queue.isNotEmpty
    currentVertex = queue.dequeue()
    for children of currentVertex
      if children.visited == false
        queue.enqueue(children)
        children.visited = true
```

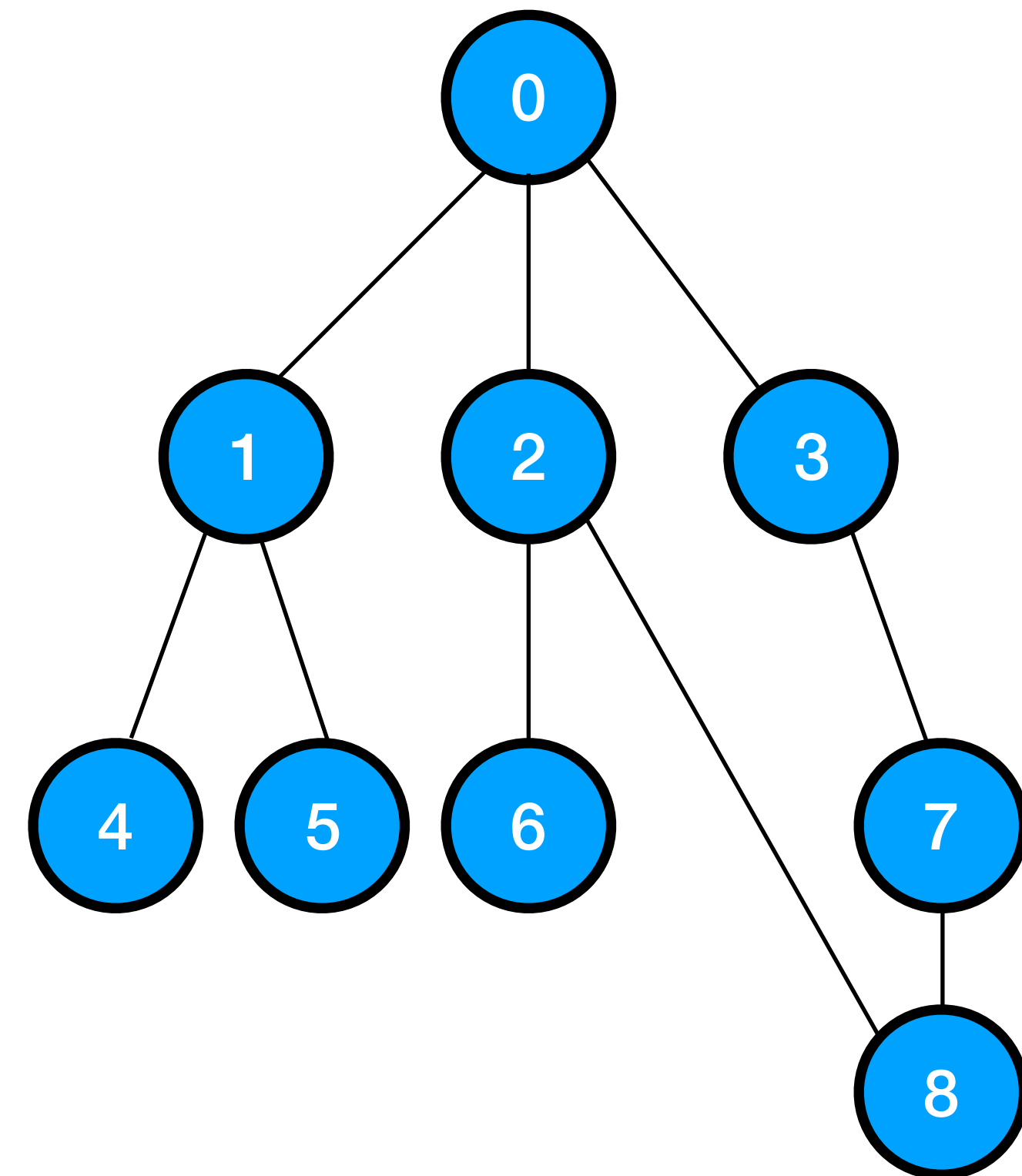
# BFS Pseudocode

```
bfs(startVertex)
  for vertex in vertices
    vertex.visited = false
  queue = new Queue
  queue.enqueue(startVertex)
  startVertex.visited = true

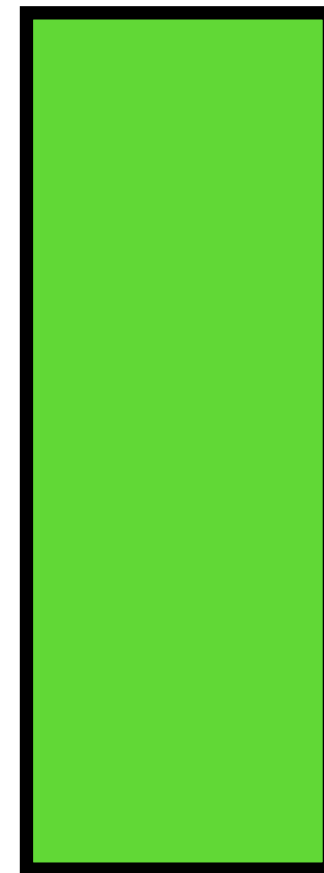
  while queue.isNotEmpty
    currentVertex = queue.dequeue()
    for children of currentVertex
      if children.visited == false
        queue.enqueue(children)
        children.visited = true
```

# Depth First Search (DFS)

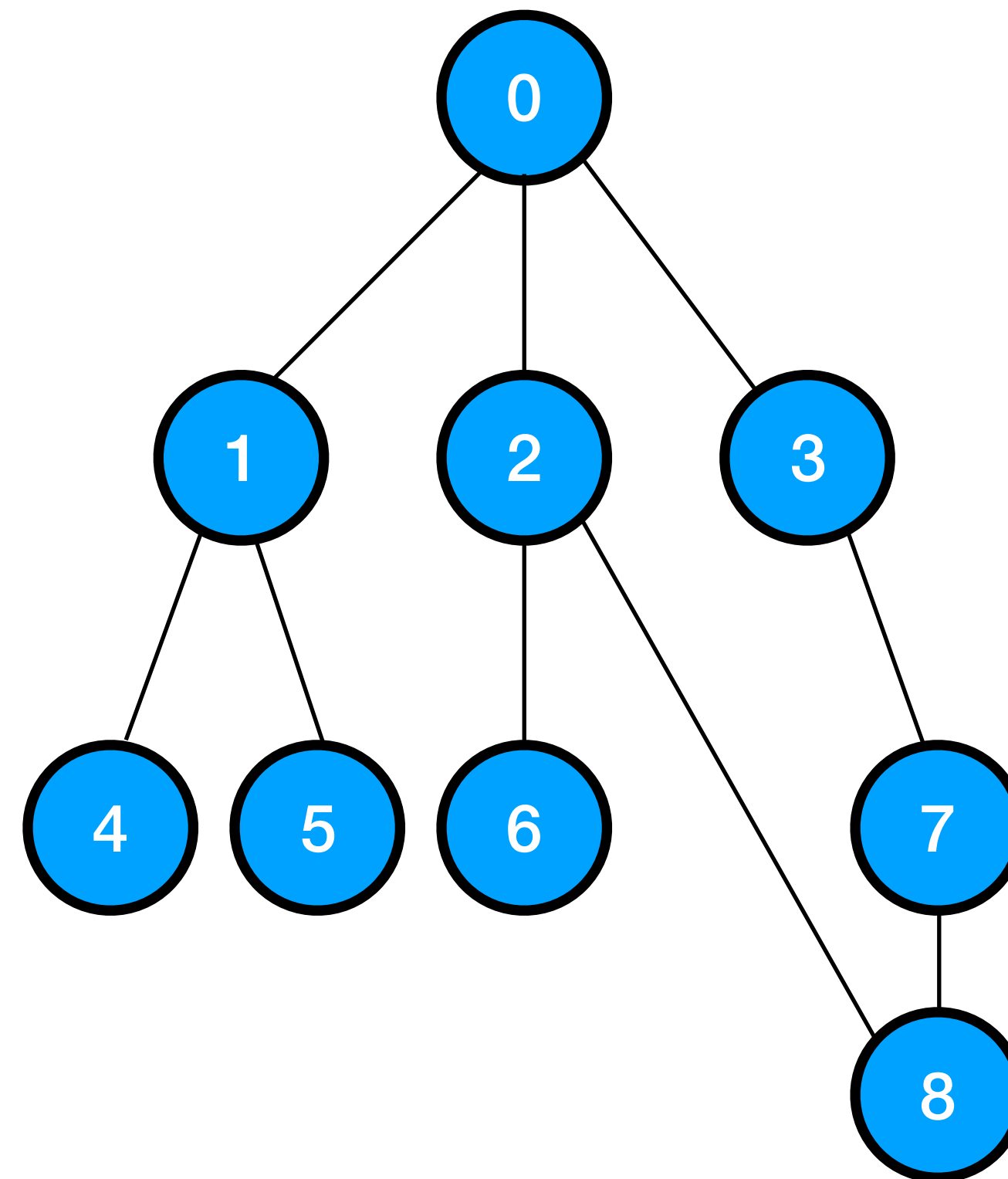
- Traverse into children before sibling/ neighboring.
- Implement using stack.



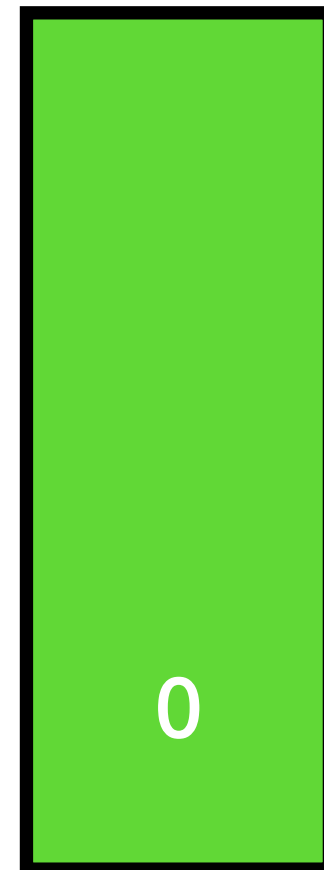
# Depth First Search (DFS)



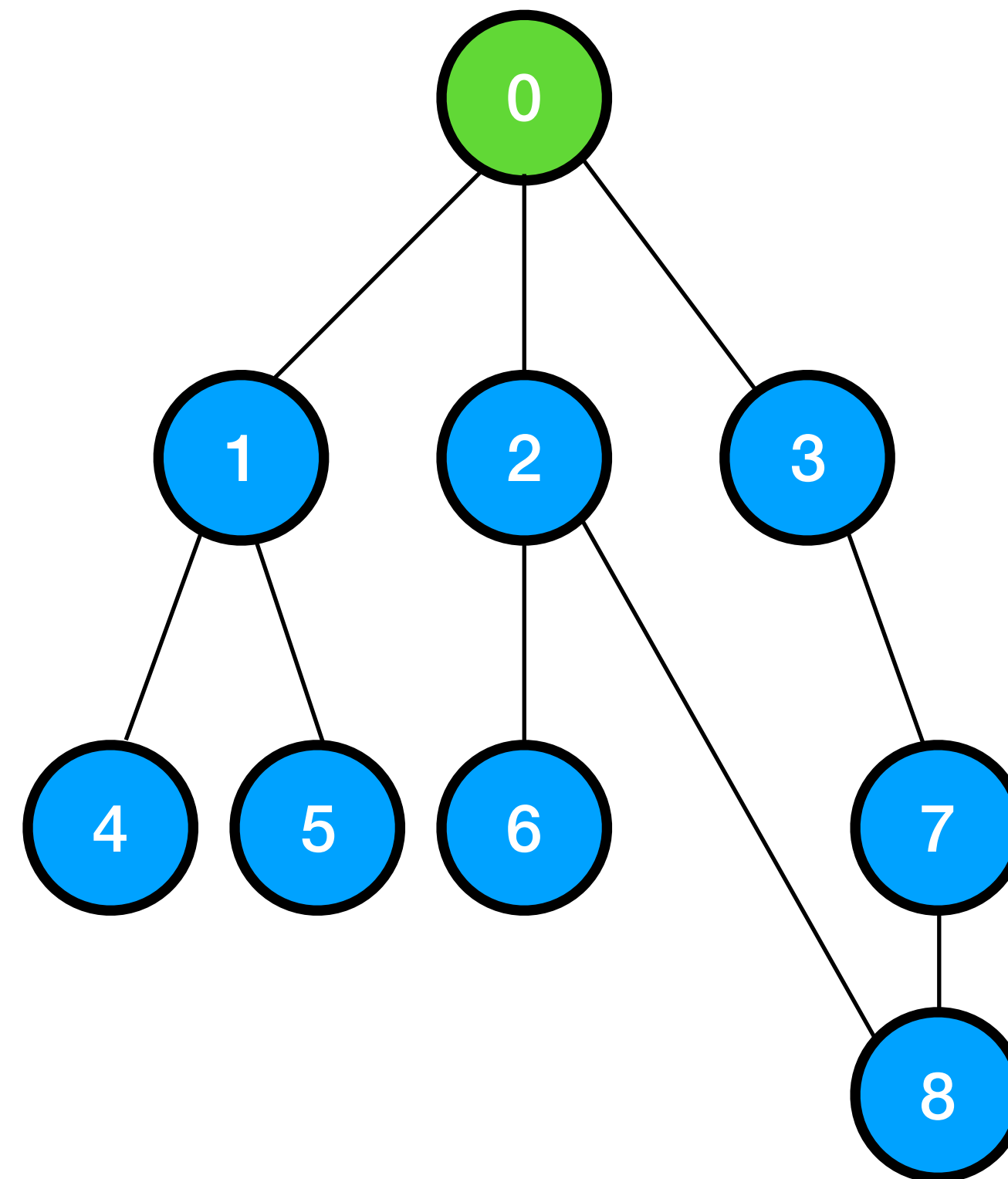
Search order :



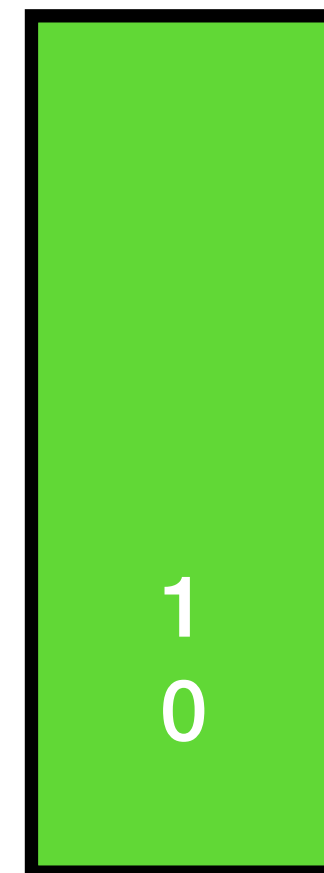
# Depth First Search (DFS)



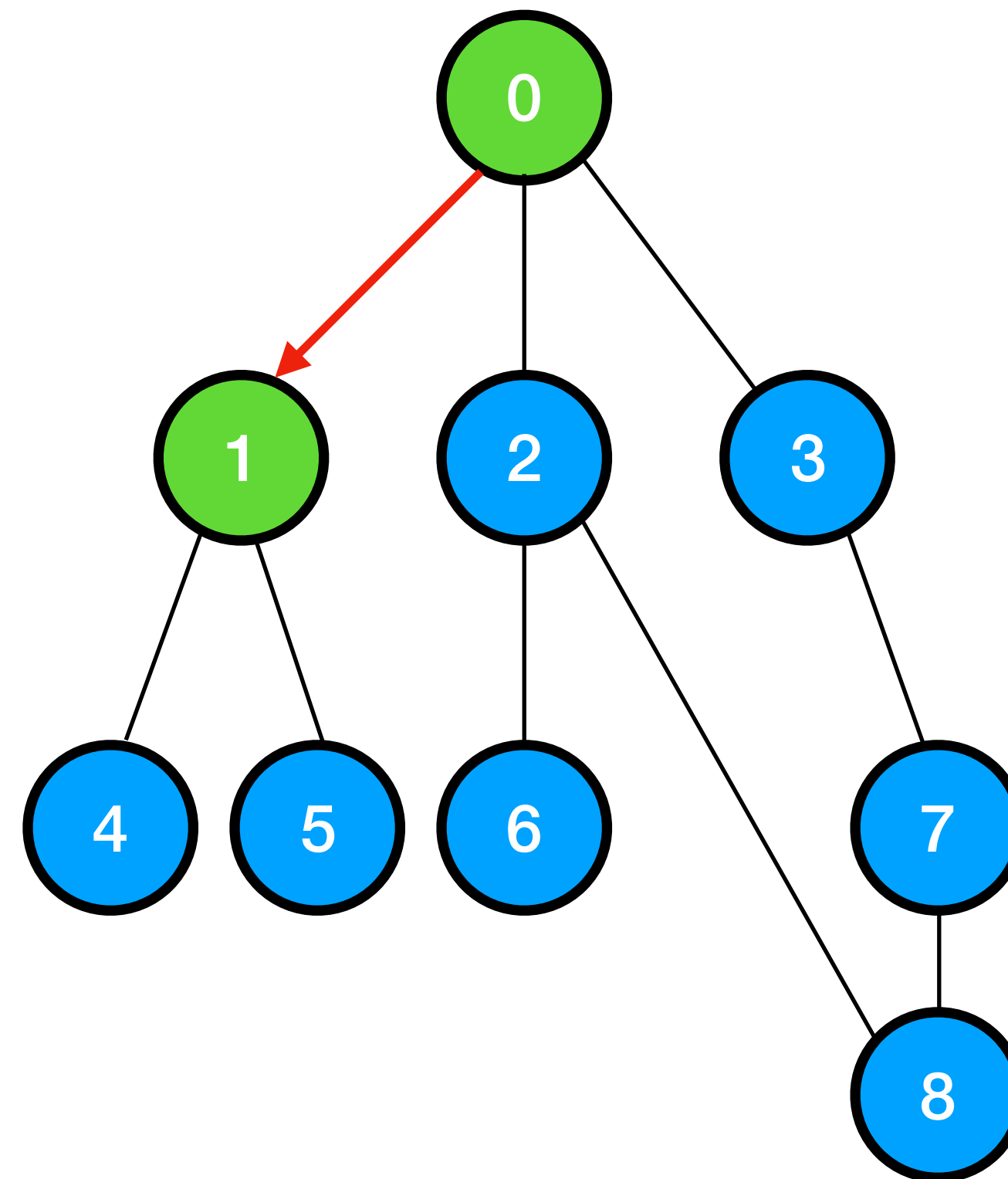
Search order : 0



# Depth First Search (DFS)

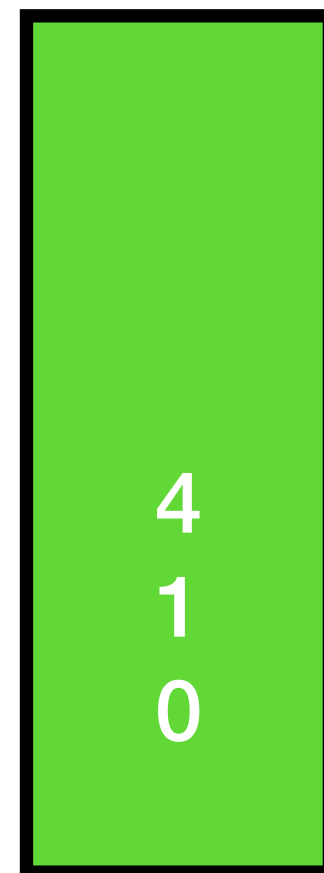


Search order : 0 1

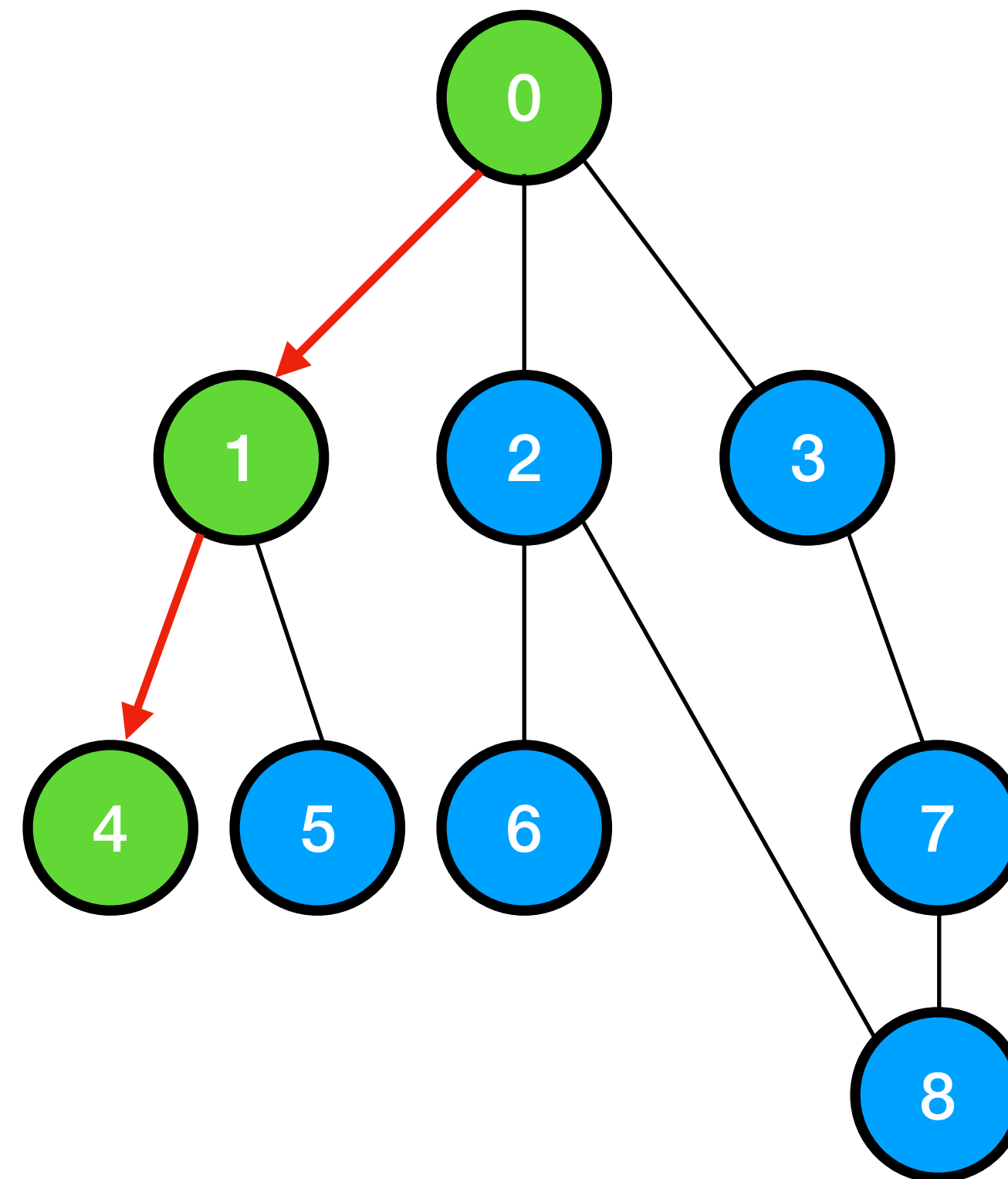




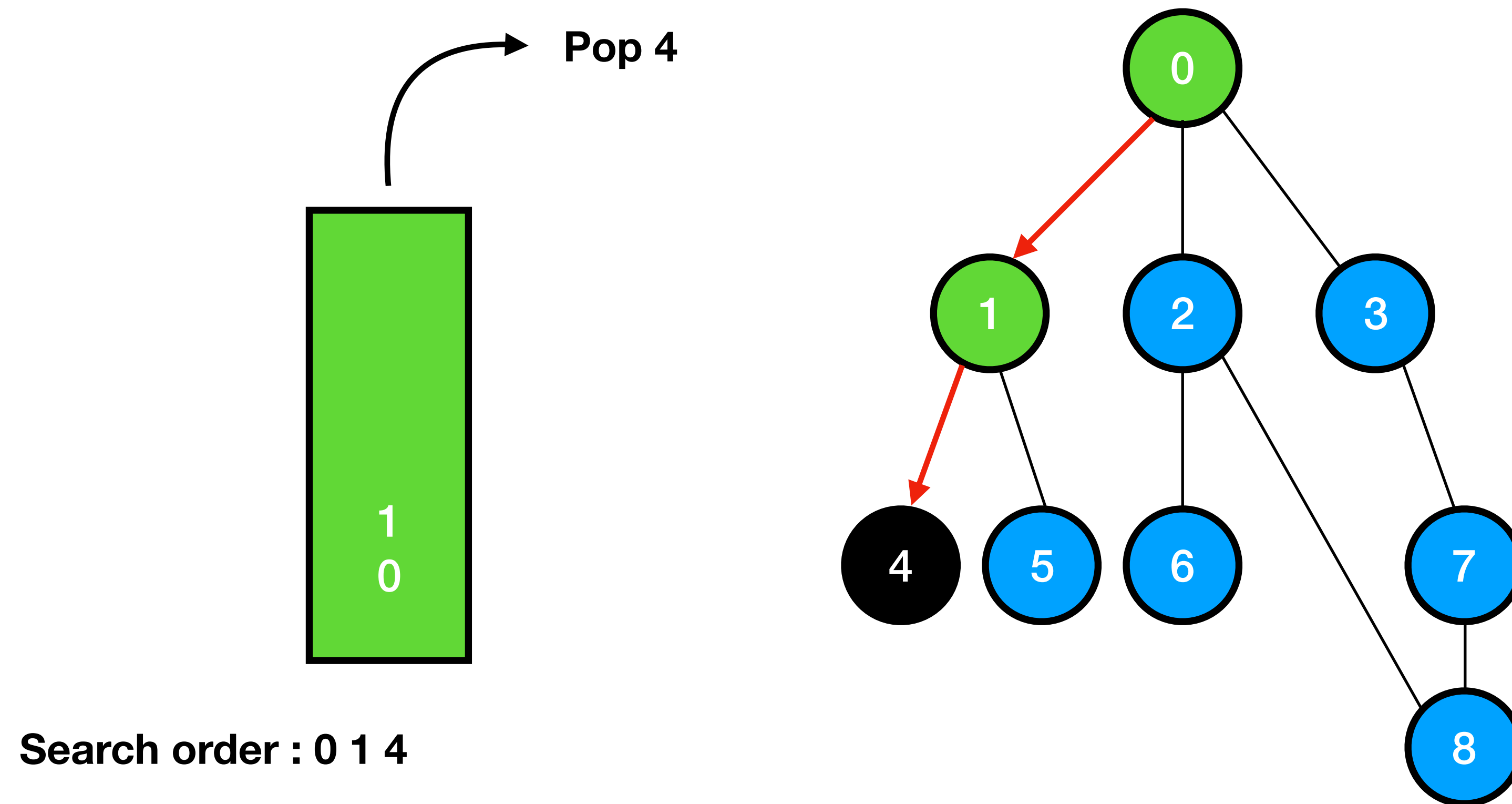
# Depth First Search (DFS)



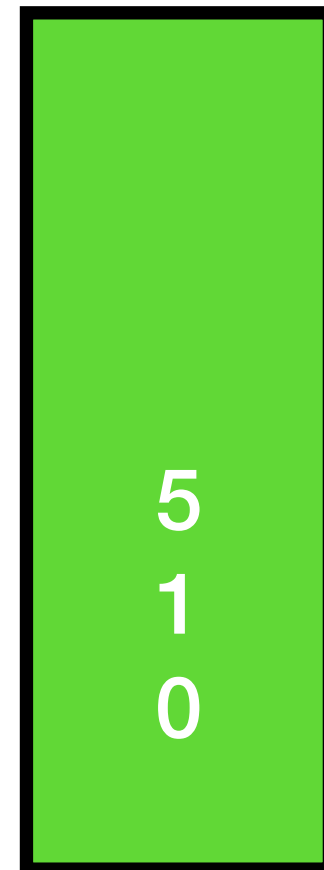
Search order : 0 1 4



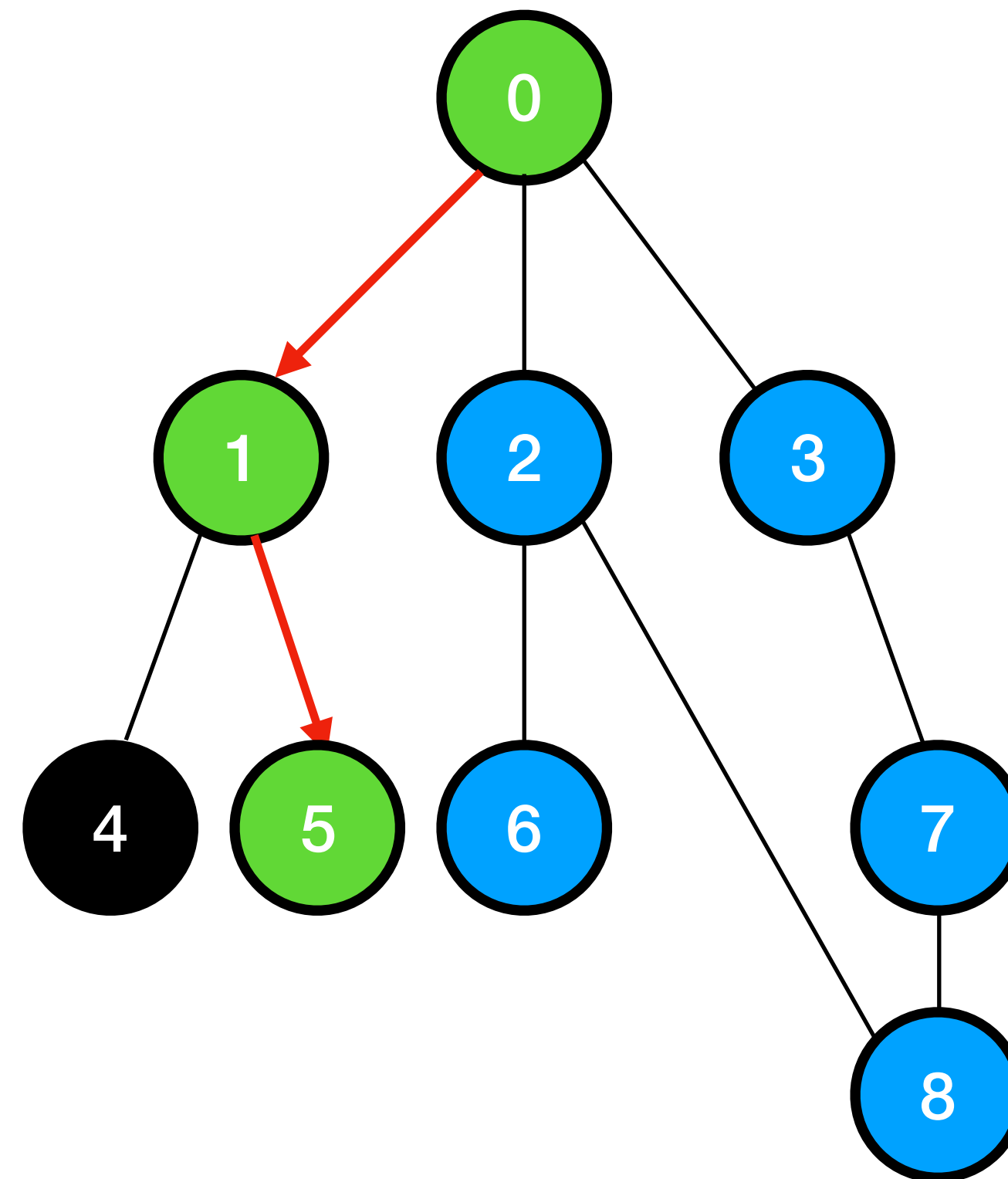
# Depth First Search (DFS)



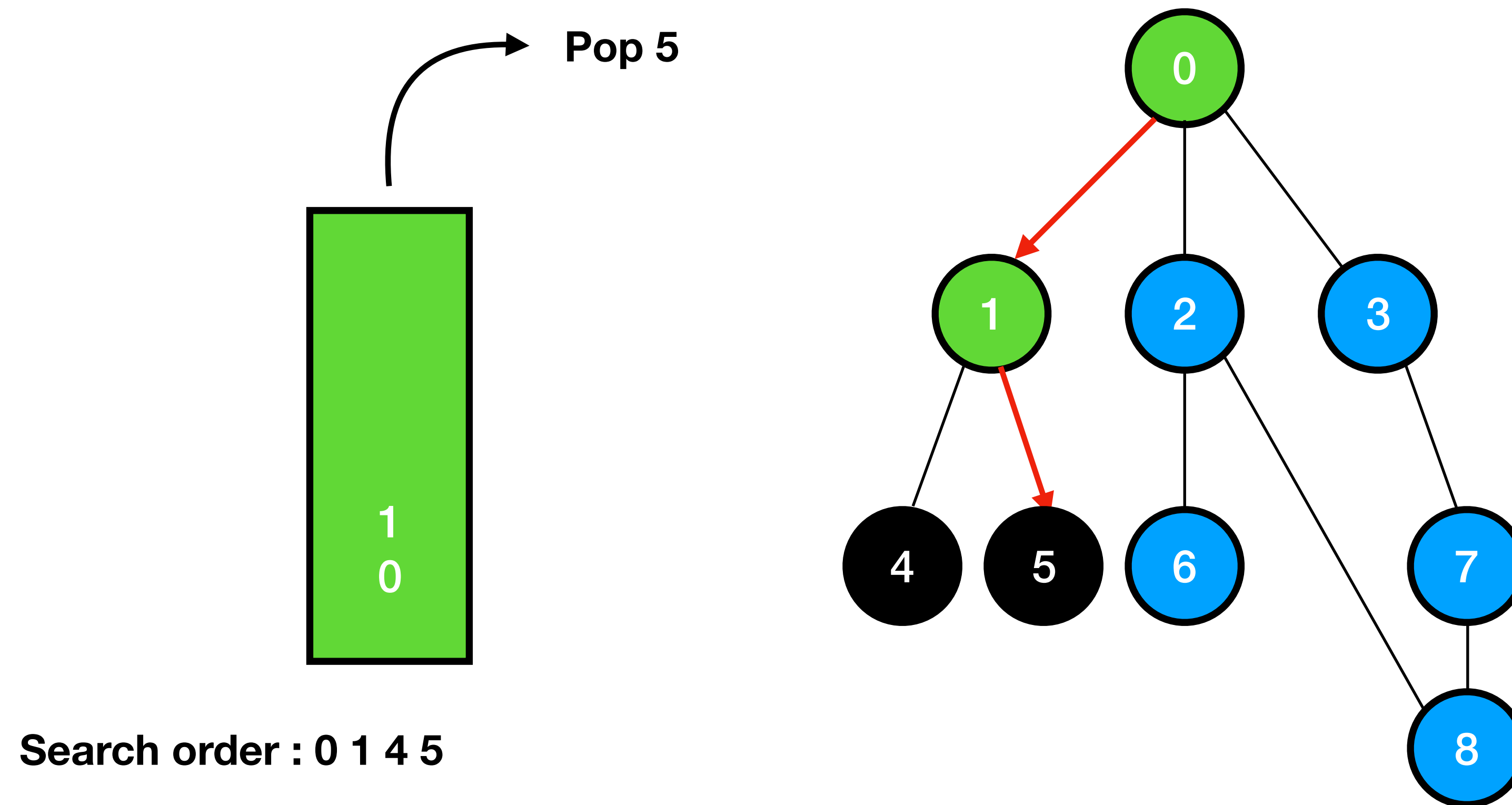
# Depth First Search (DFS)



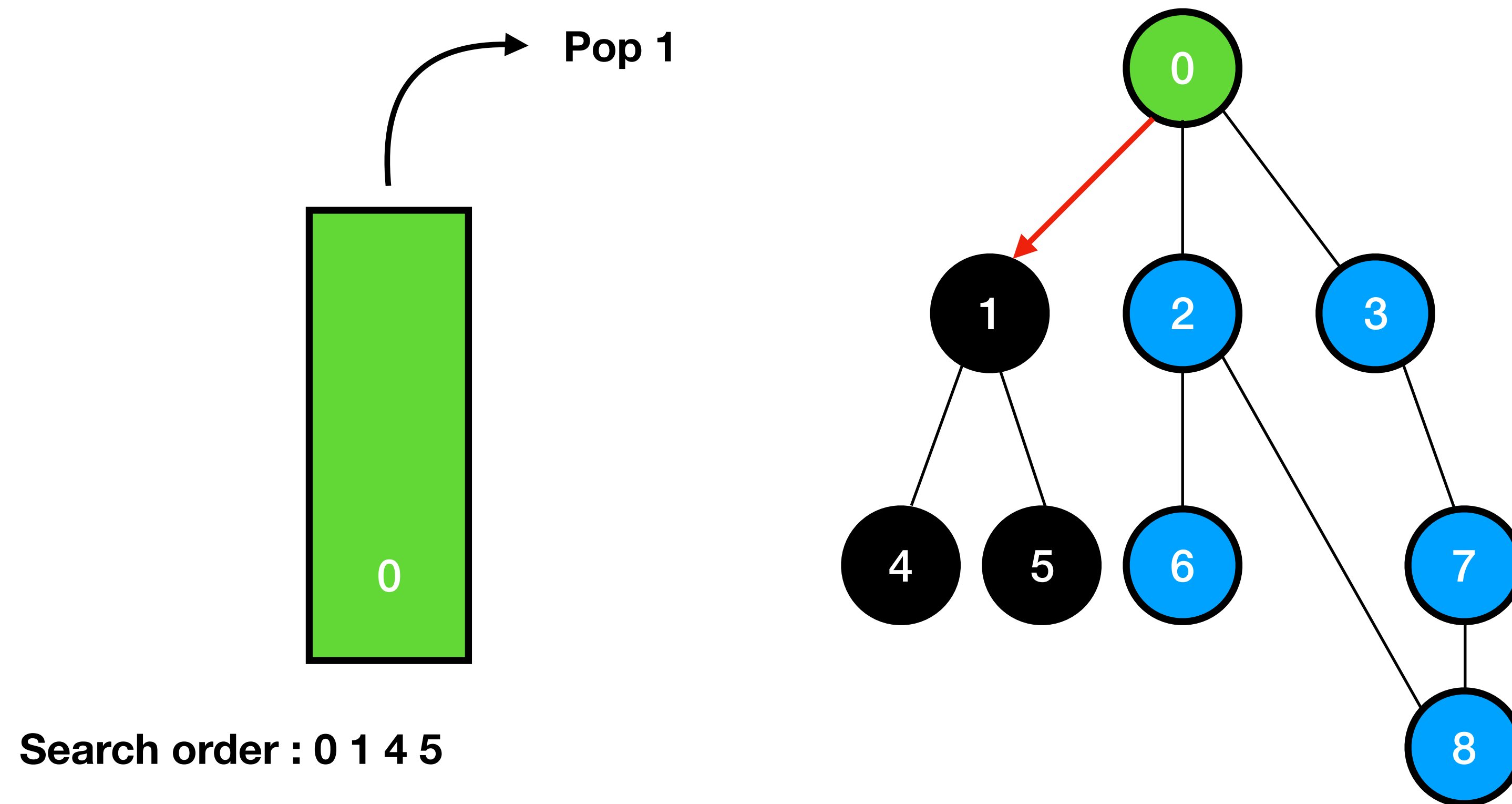
Search order : 0 1 4 5



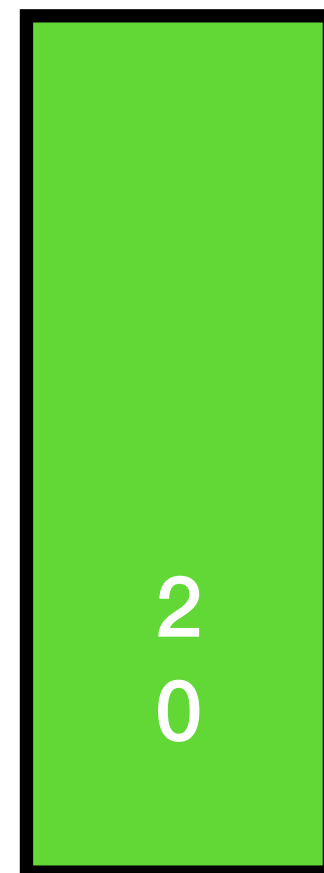
# Depth First Search (DFS)



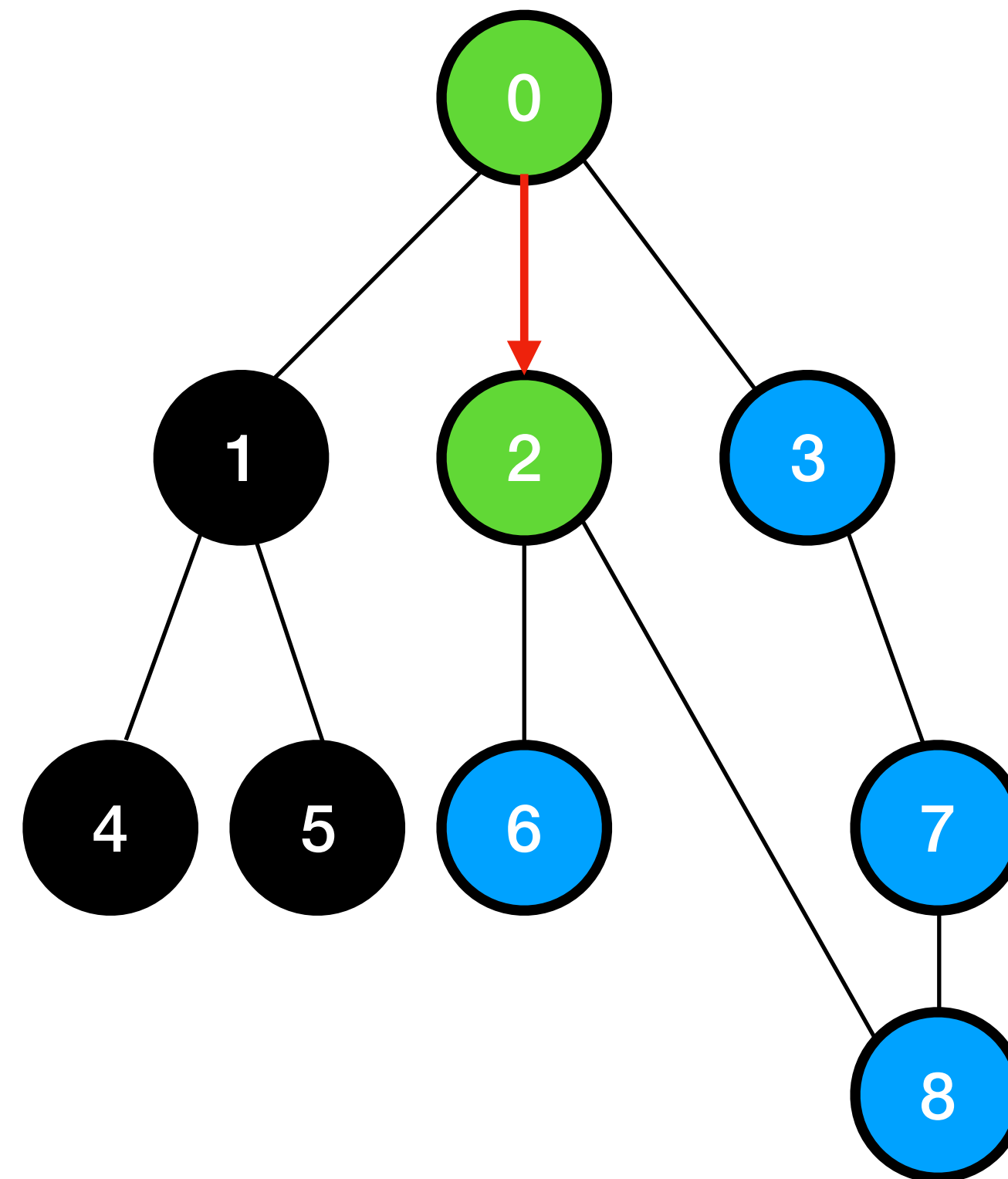
# Depth First Search (DFS)



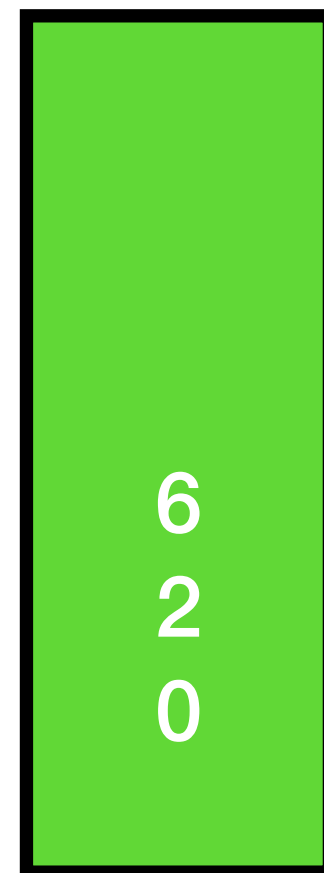
# Depth First Search (DFS)



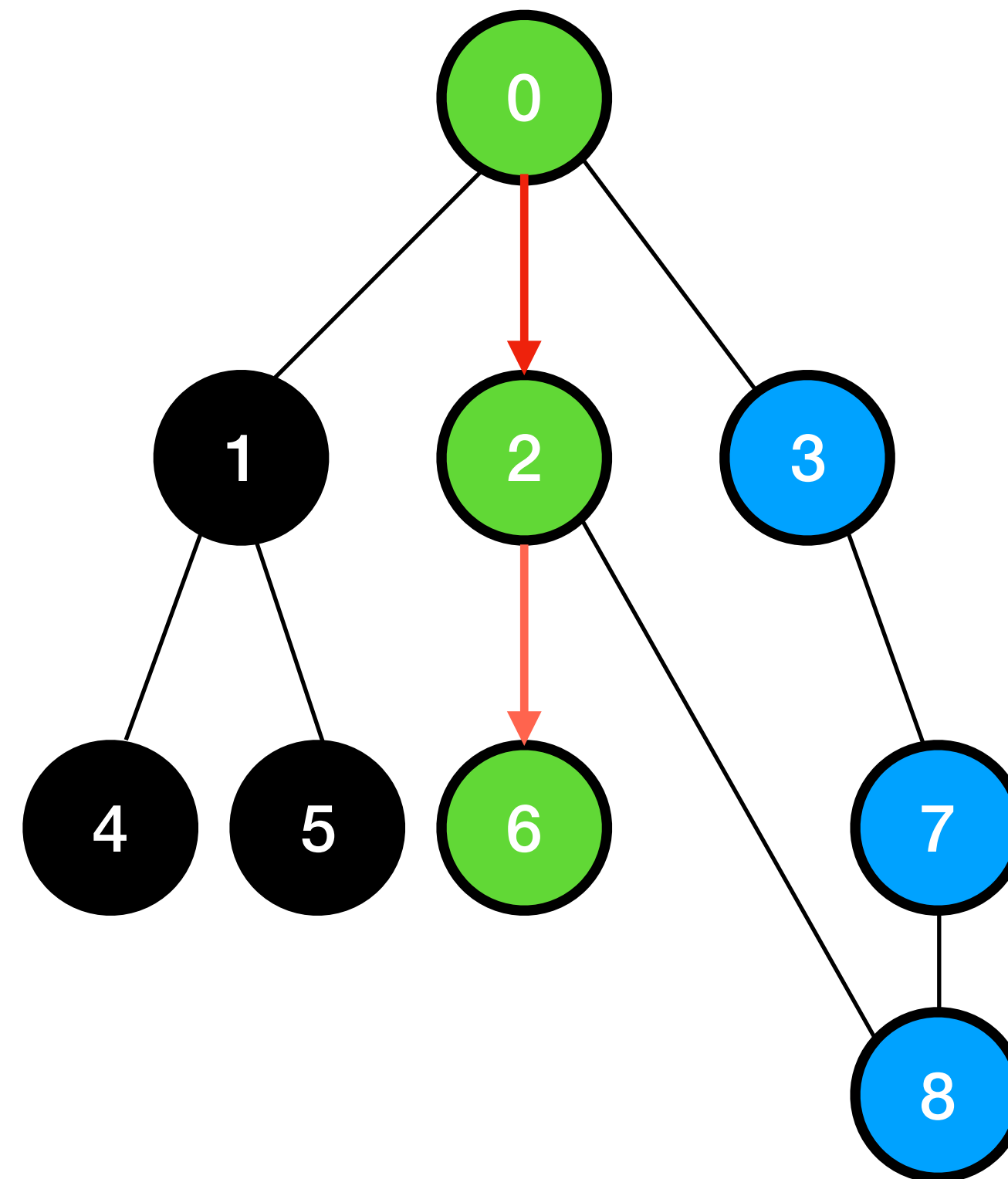
Search order : 0 1 4 5 2



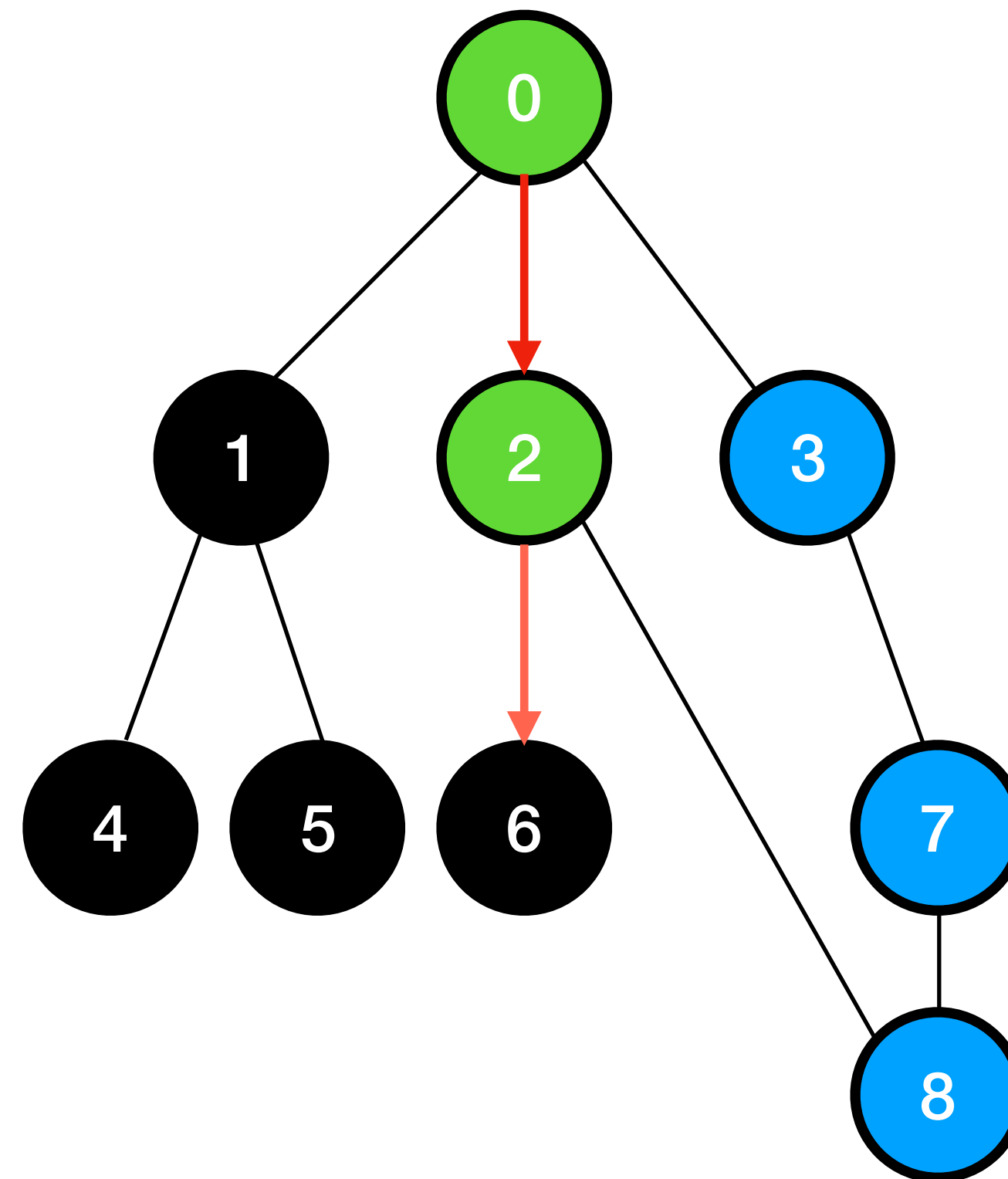
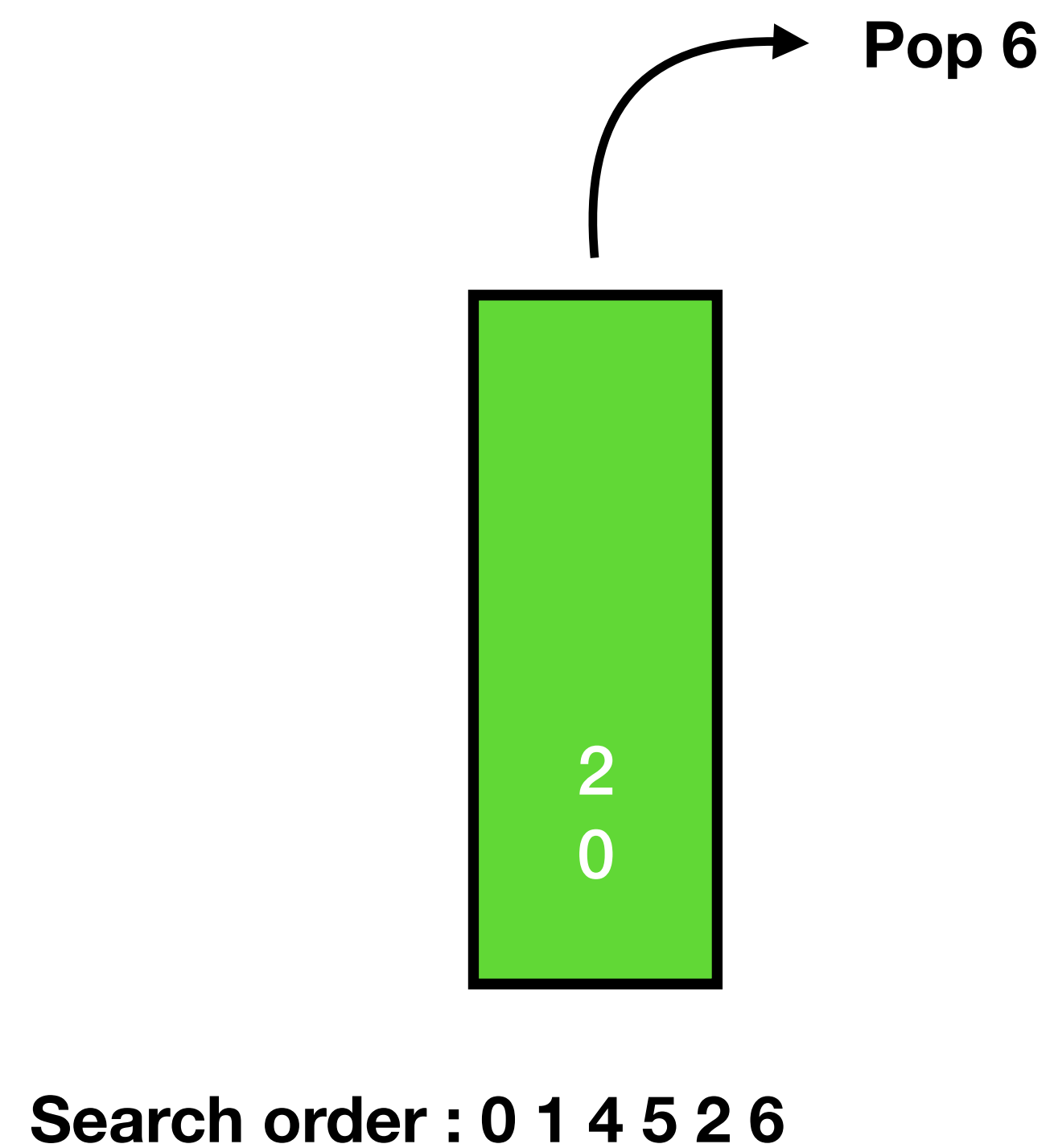
# Depth First Search (DFS)



Search order : 0 1 4 5 2 6

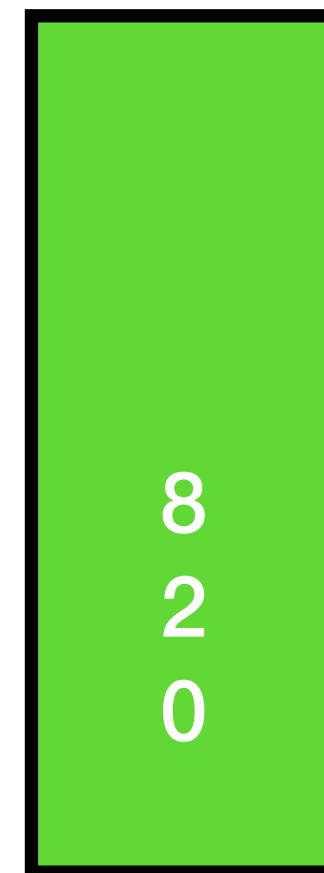


# Depth First Search (DFS)

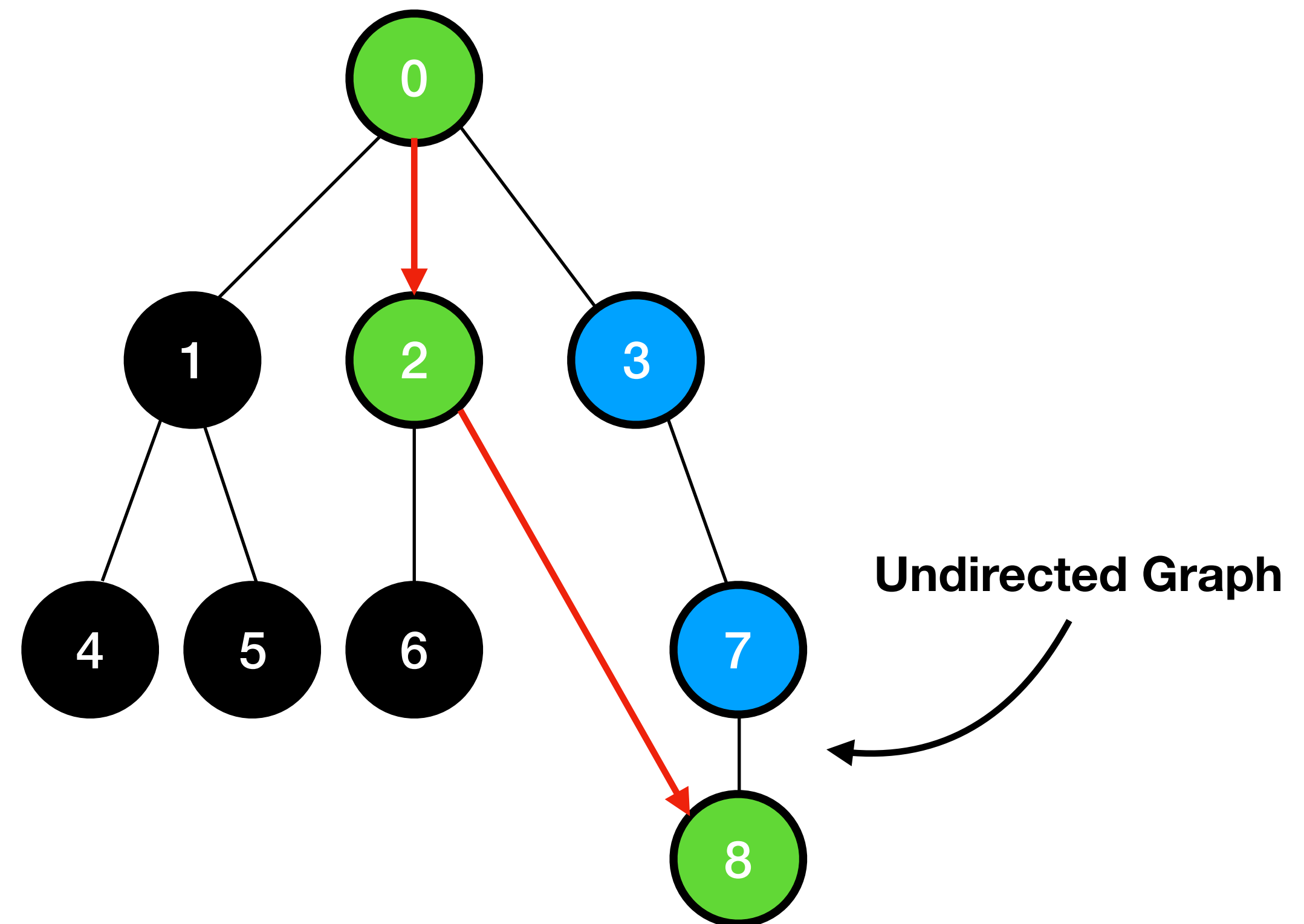




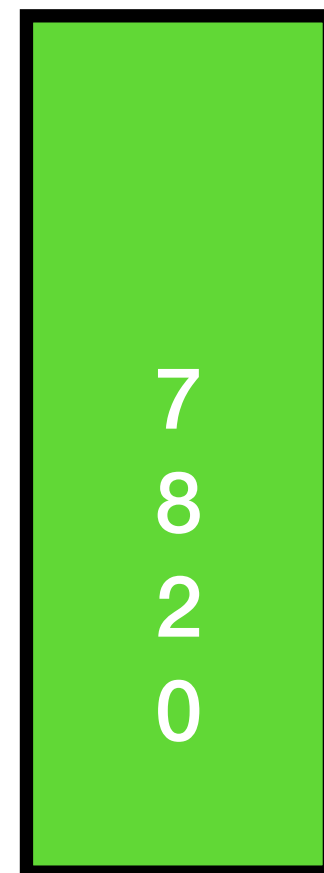
# Depth First Search (DFS)



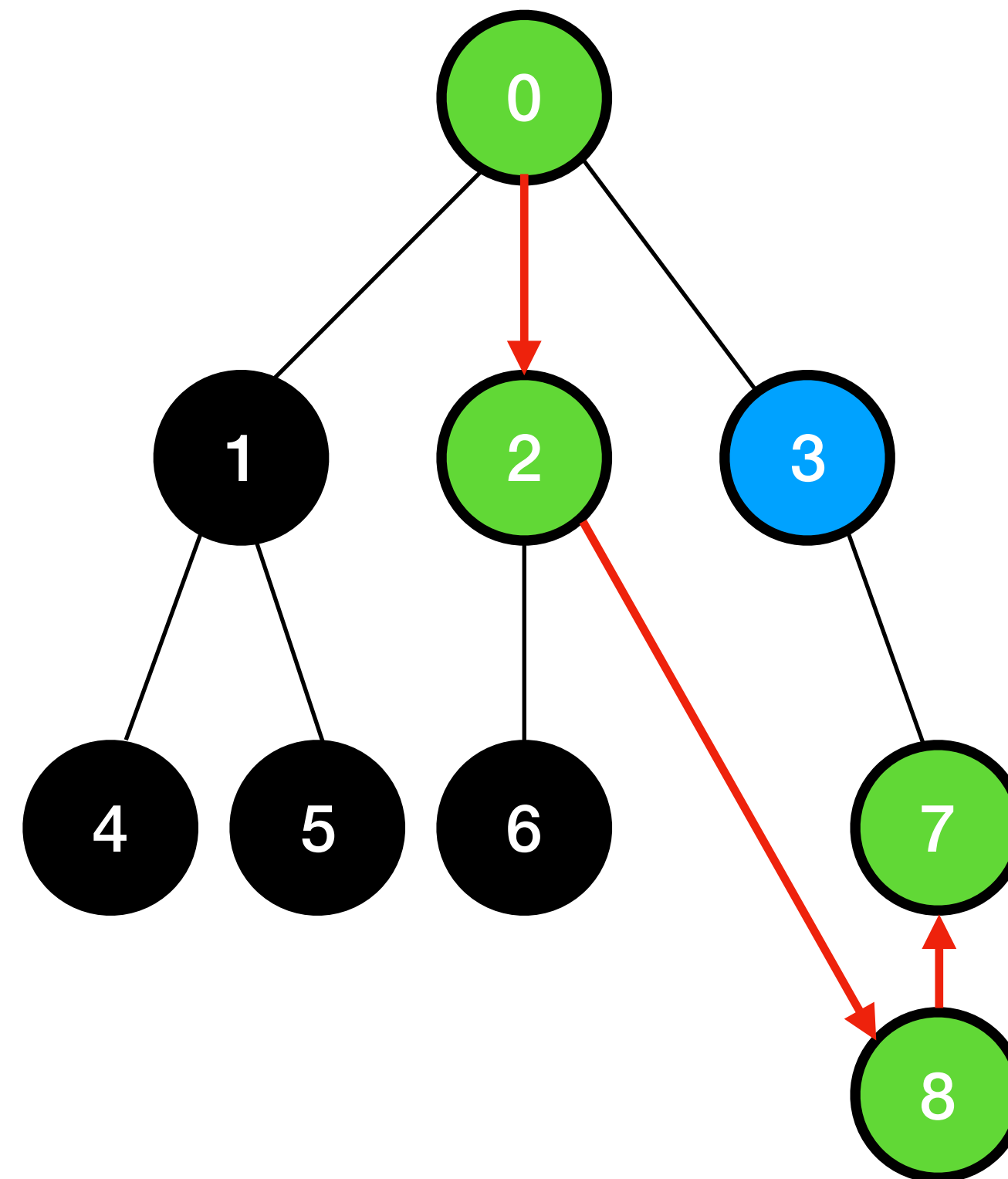
Search order : 0 1 4 5 2 6 8



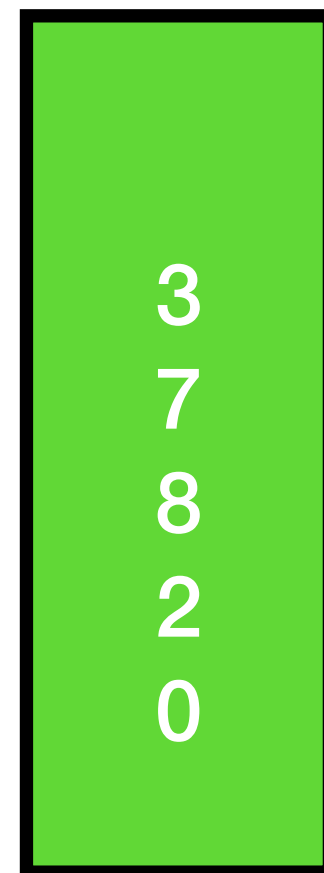
# Depth First Search (DFS)



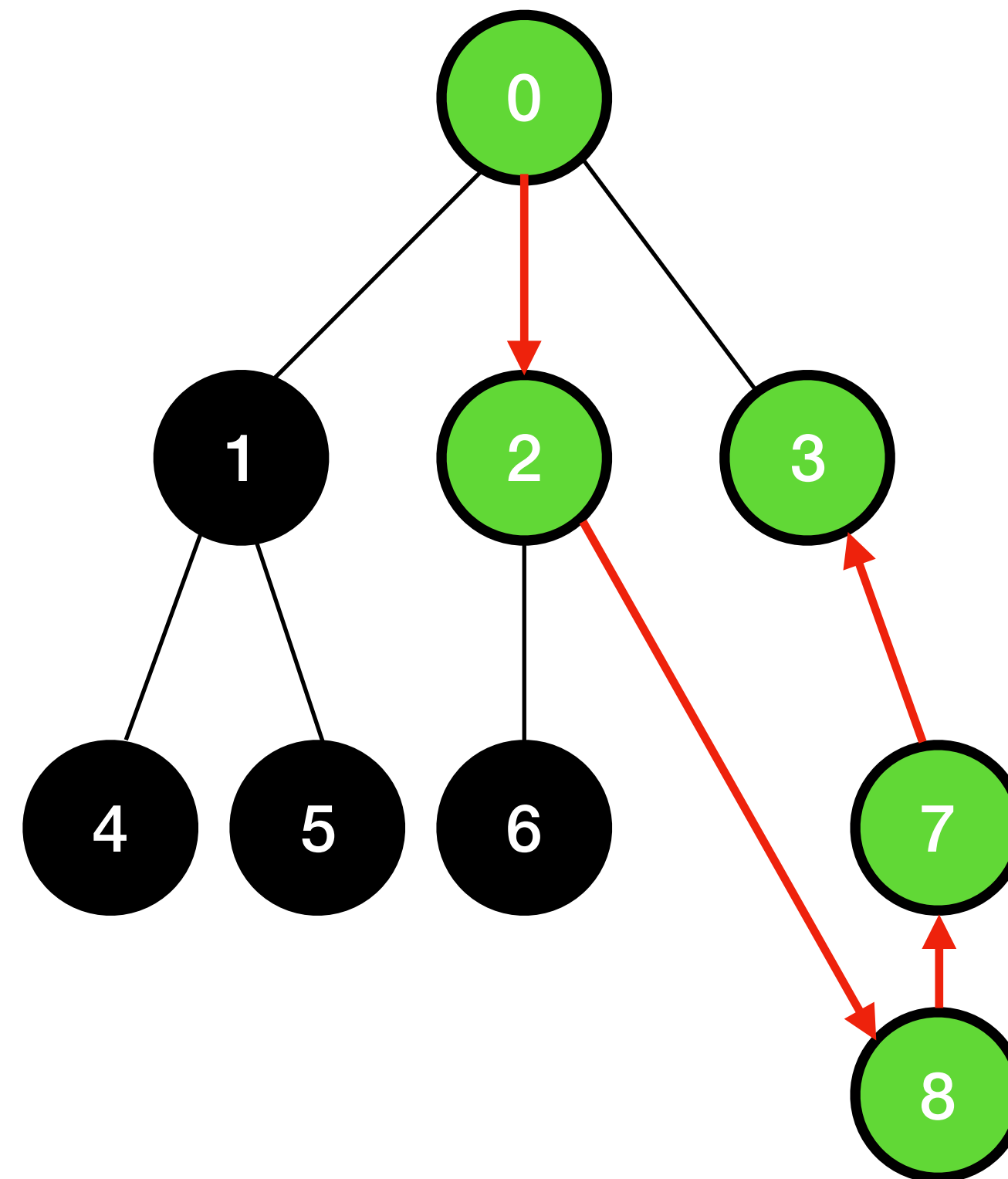
Search order : 0 1 4 5 2 6 8 7



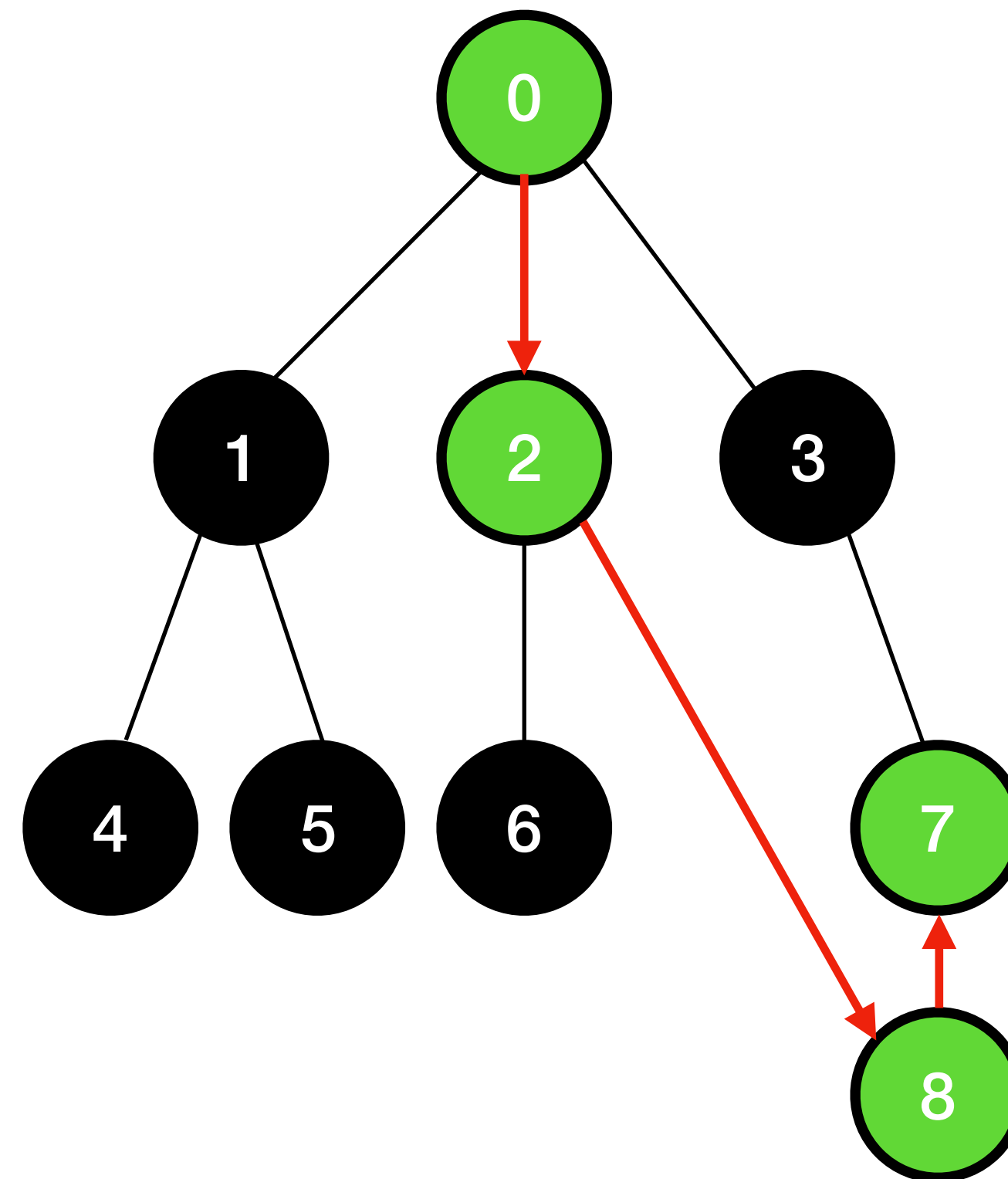
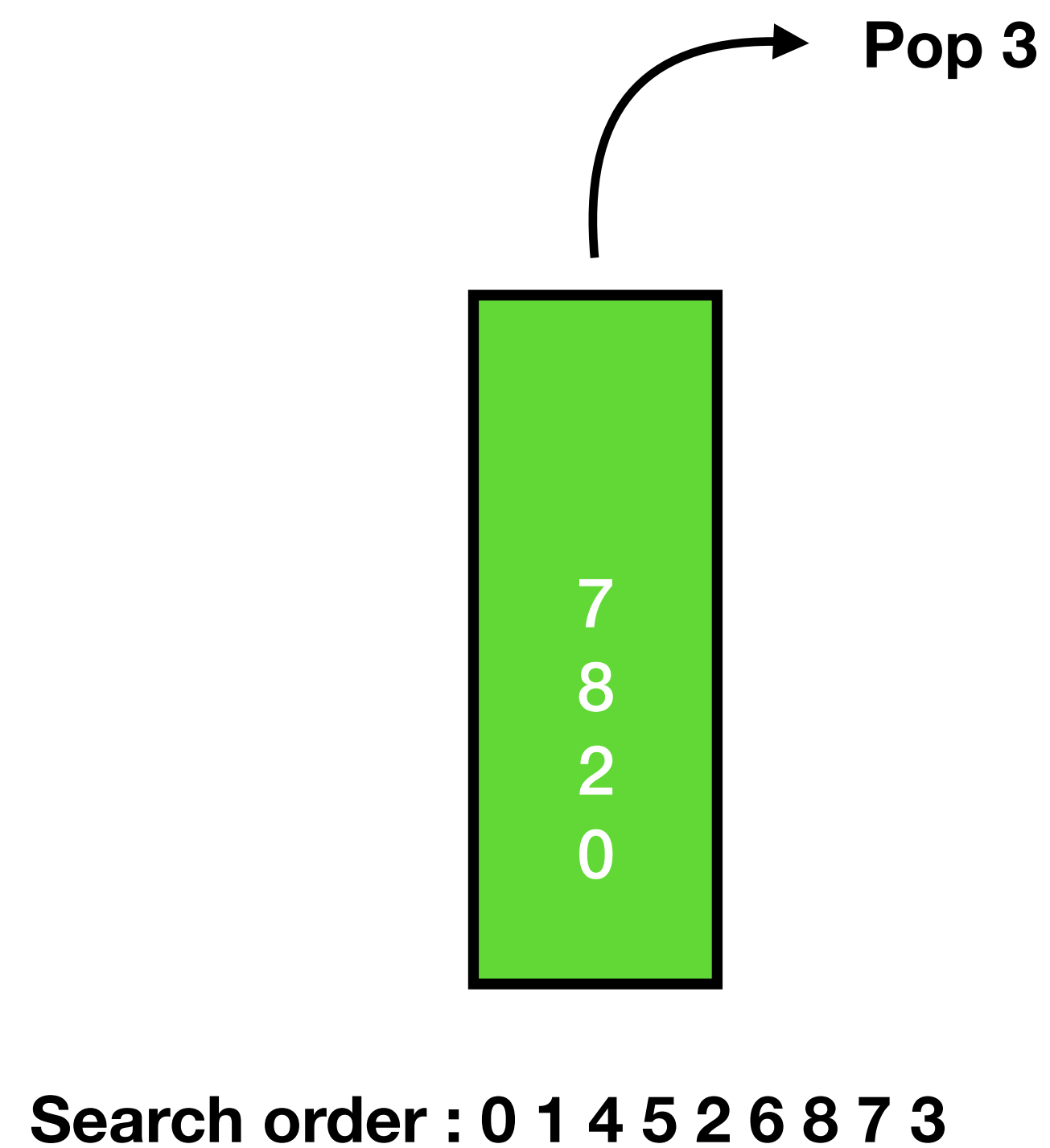
# Depth First Search (DFS)



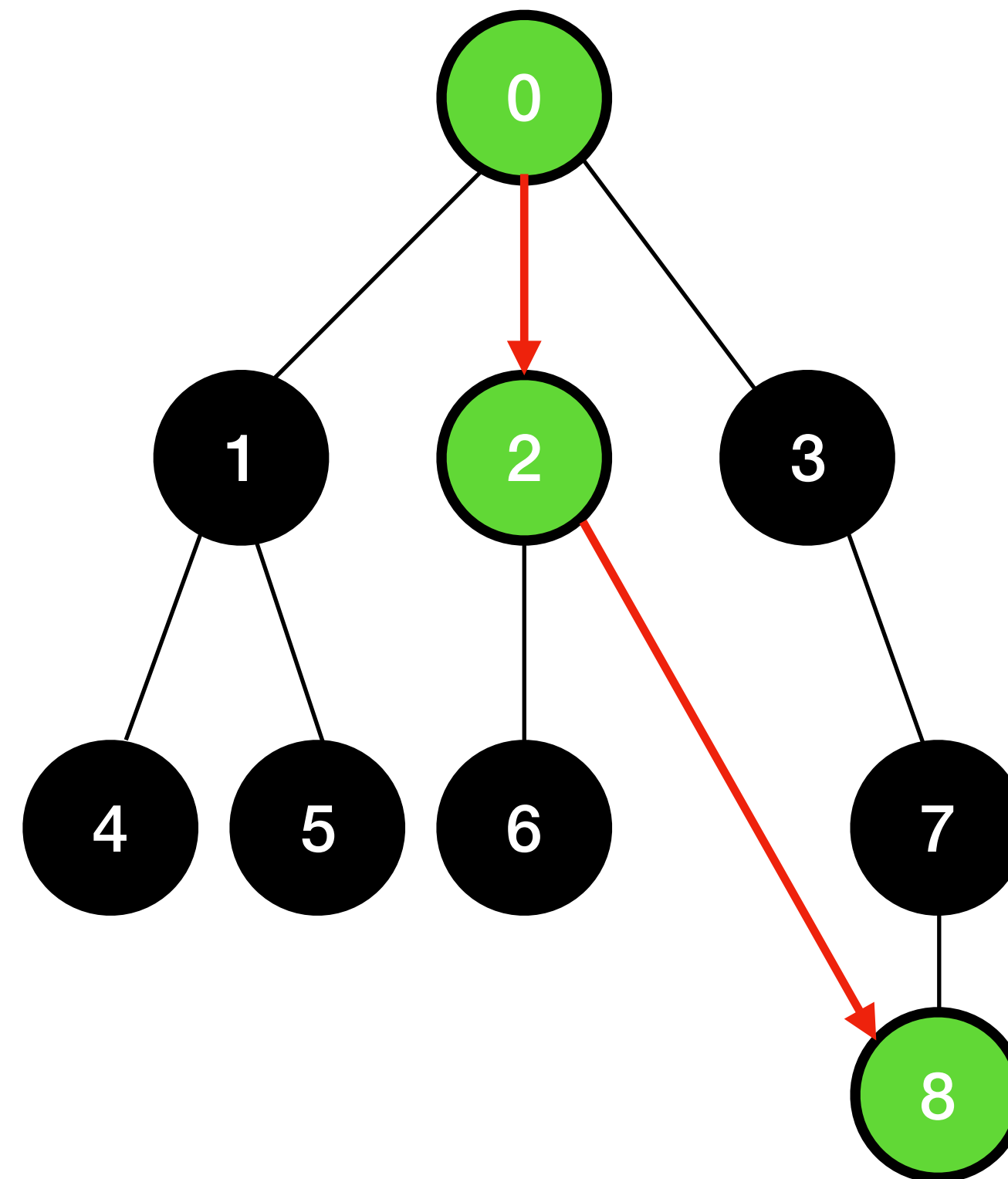
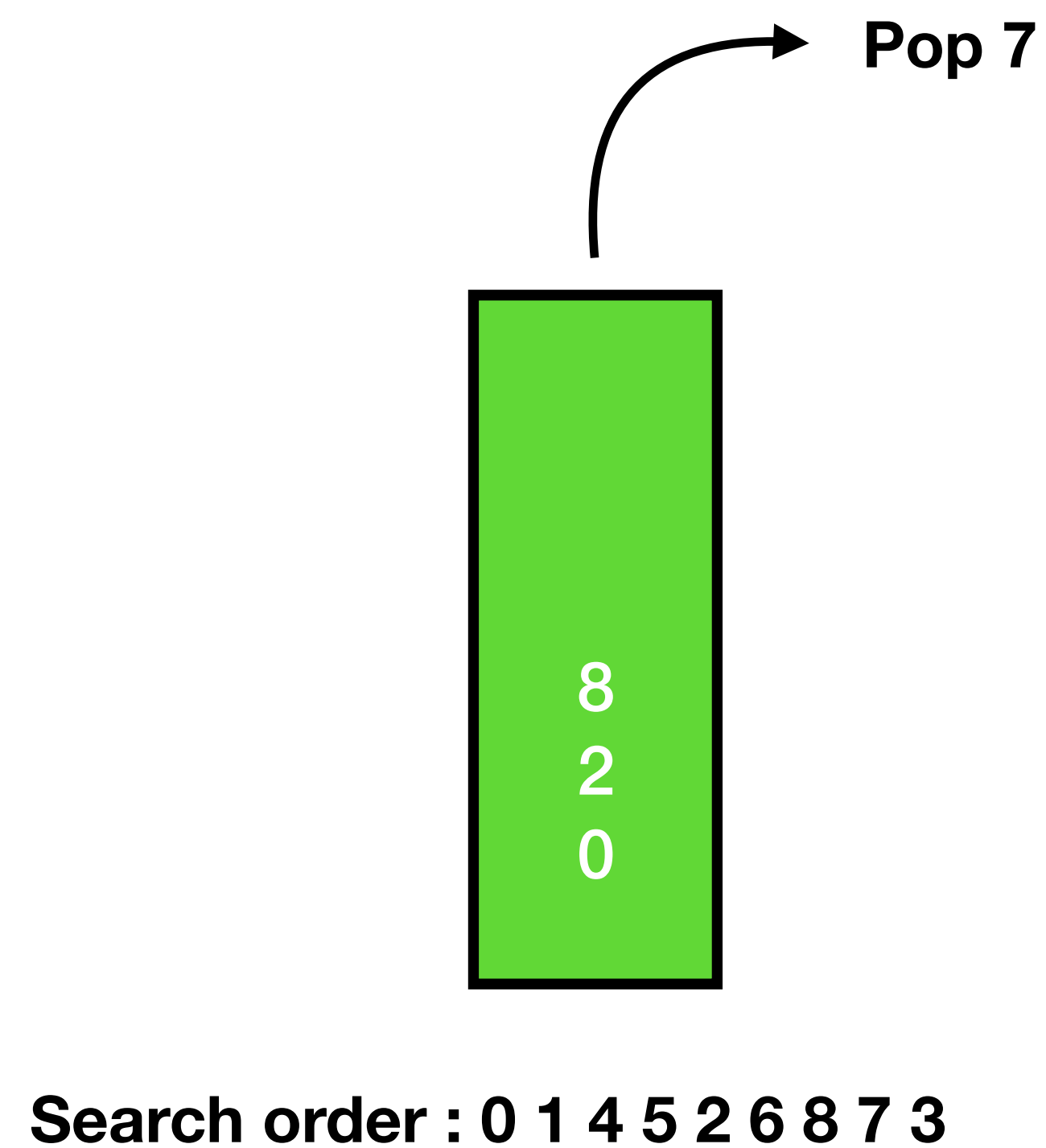
Search order : 0 1 4 5 2 6 8 7 3



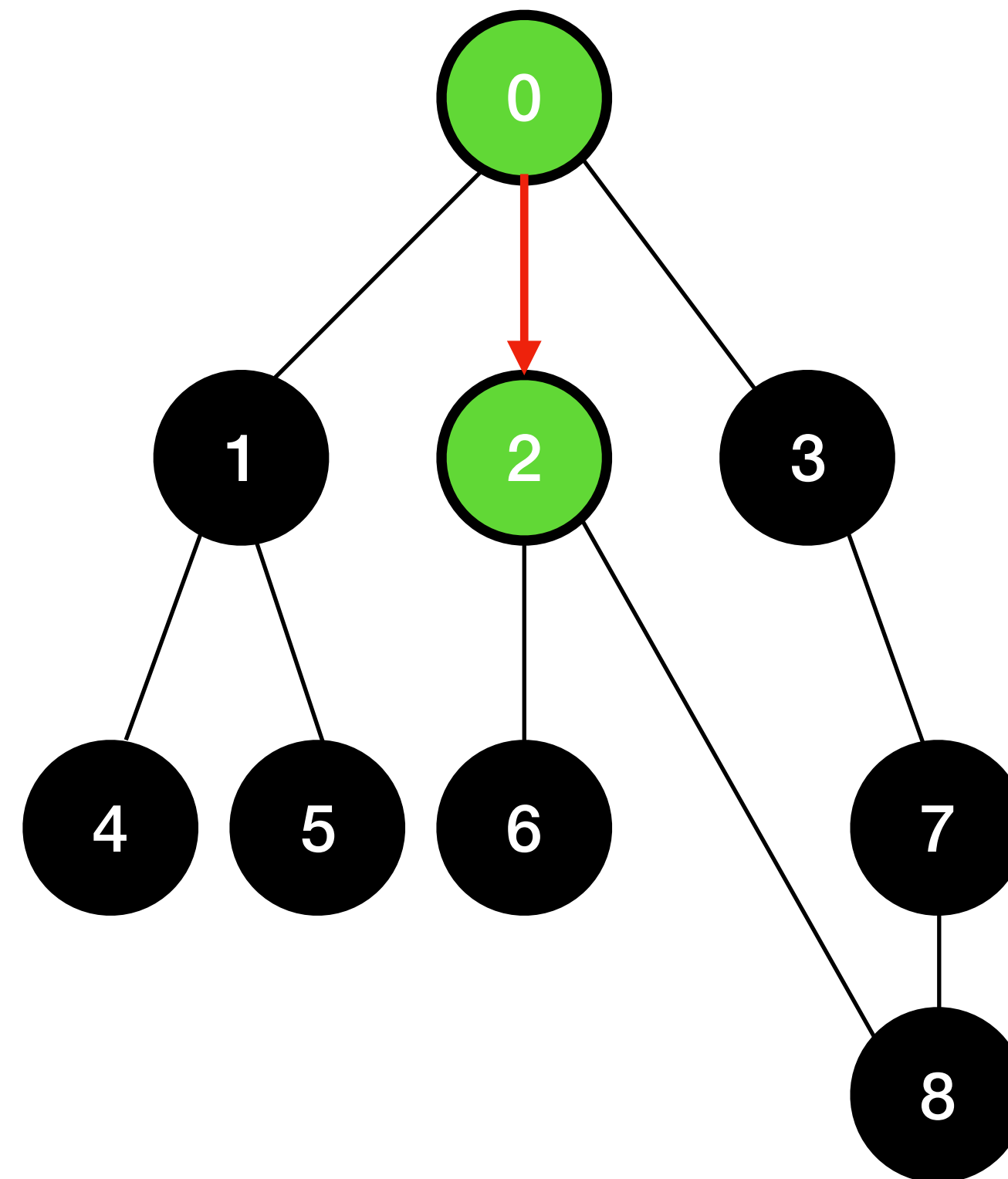
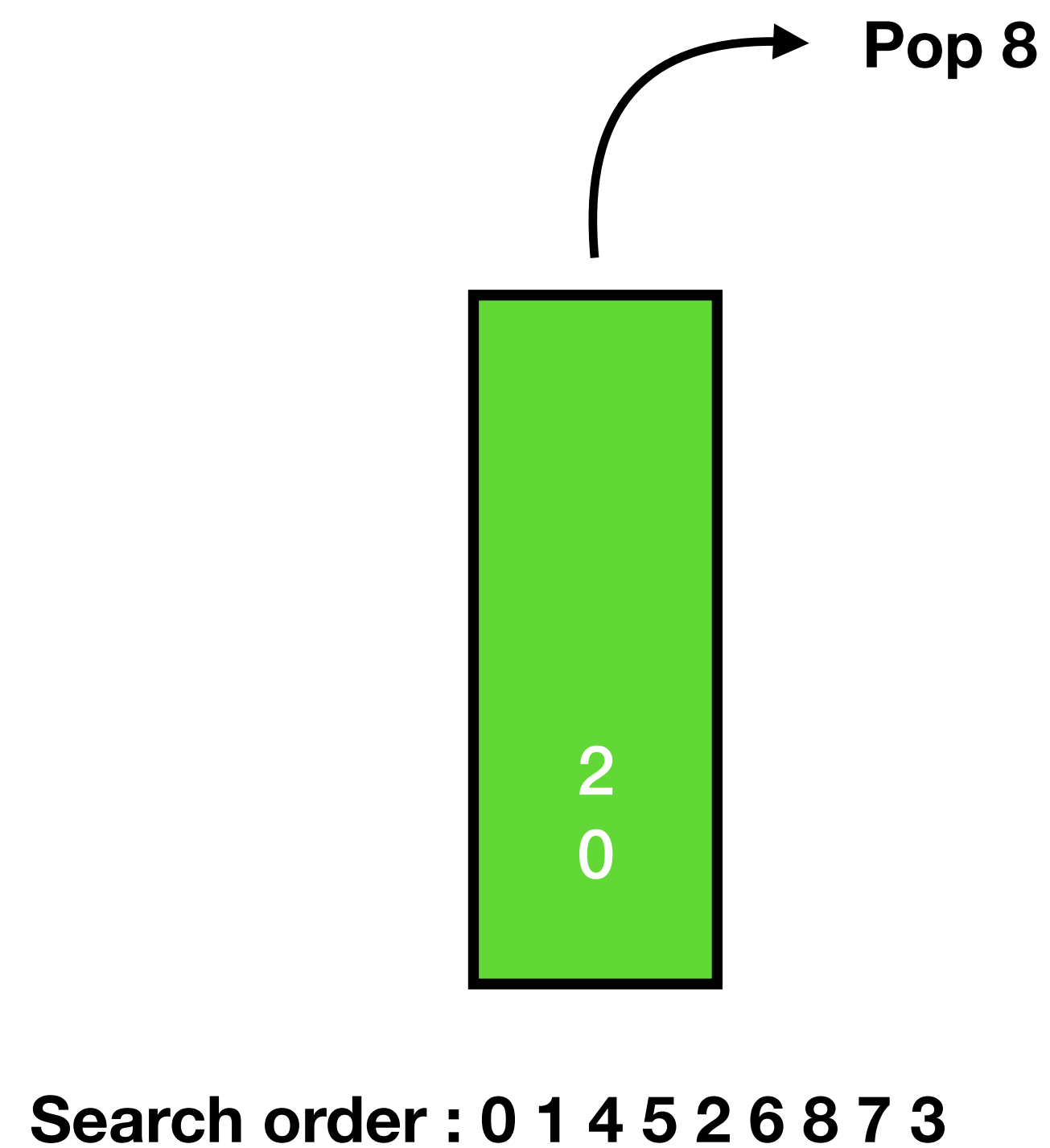
# Depth First Search (DFS)



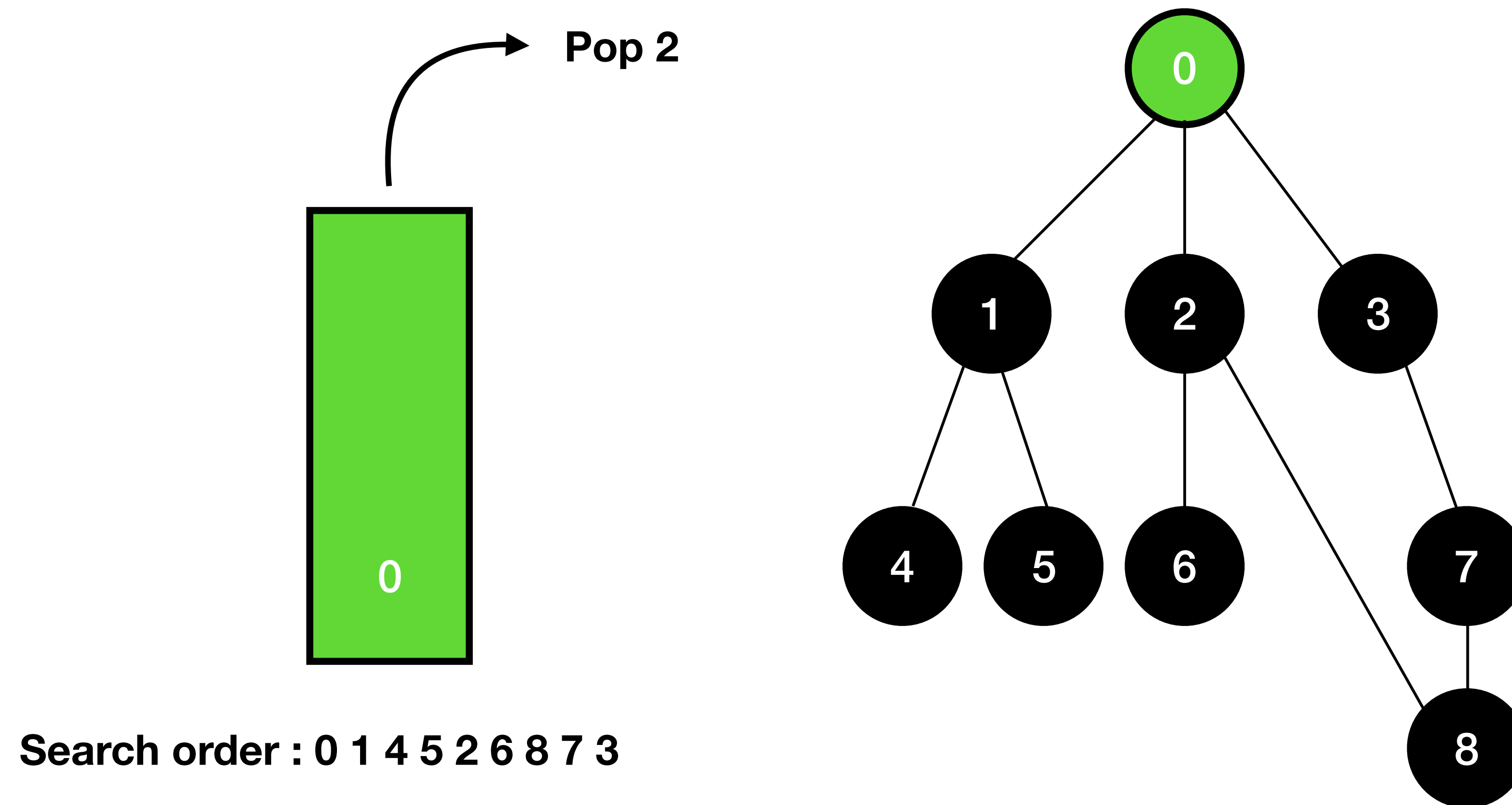
# Depth First Search (DFS)



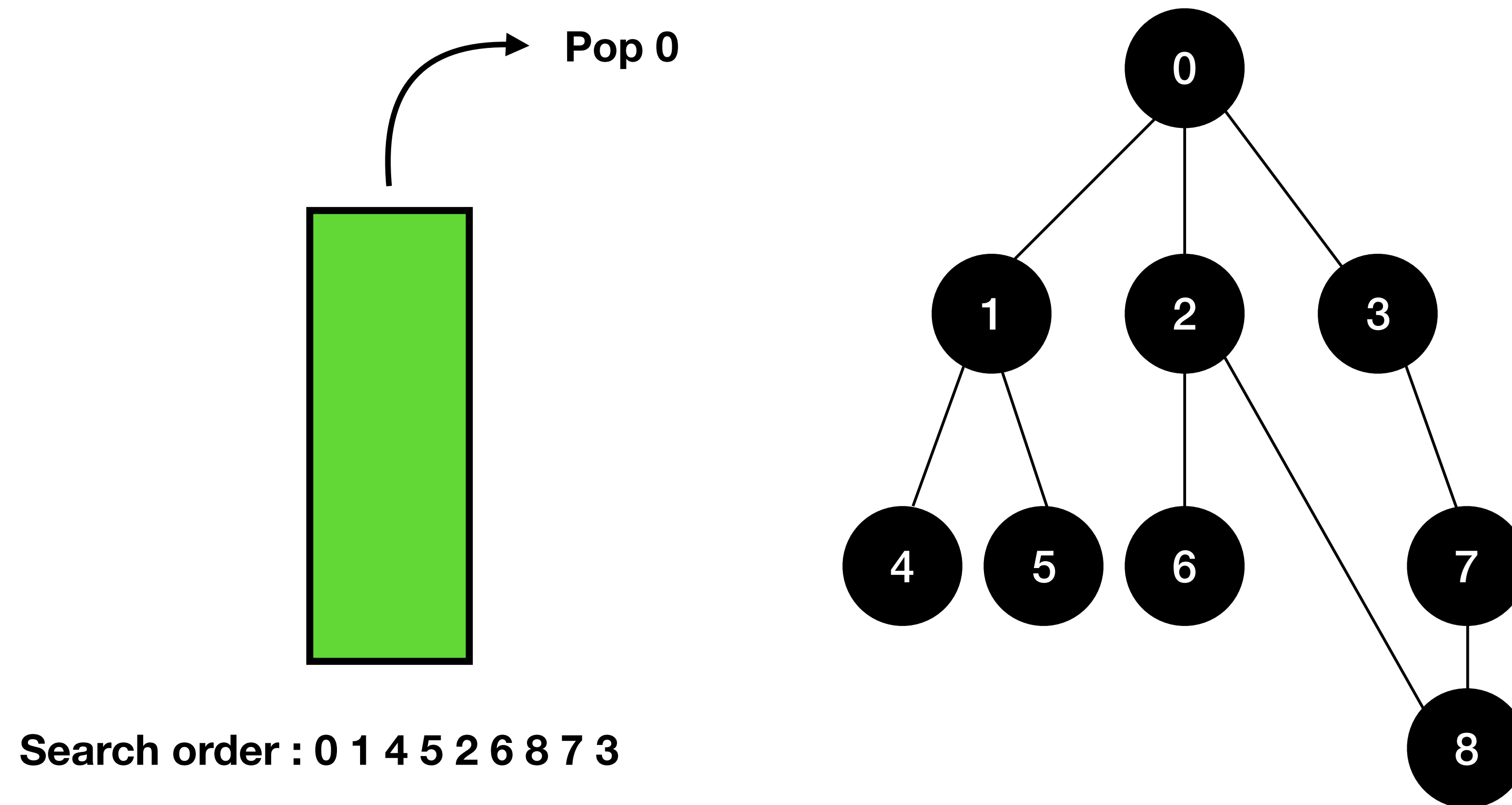
# Depth First Search (DFS)



# Depth First Search (DFS)



# Depth First Search (DFS)





# DFS Pseudocode

**dfs(startVertex)**

for vertex in vertices

    vertex.visited = false

dfsRecursive(startVertex, visitedVertices)

dfsRecursive(currentVertex, visitedVertices)

    currentVertex.visited = true

for children of currentVertex

    if children.visited == false

        dfsRecursive(children, visitedVertices)

# DFS Pseudocode

```
dfs(startVertex)
```

```
  for vertex in vertices
```

```
    vertex.visited = false
```

```
  dfsRecursive(startVertex, visitedVertices)
```

```
dfsRecursive(currentVertex, visitedVertices)
```

```
  currentVertex.visited = true
```

```
  for children of currentVertex
```

```
    if children.visited == false
```

```
      dfsRecursive(children, visitedVertices)
```

# DFS Pseudocode

```
dfs(startVertex)
  for vertex in vertices
    vertex.visited = false
  dfsRecursive(startVertex, visitedVertices)
```

```
dfsRecursive(currentVertex, visitedVertices)
  currentVertex.visited = true

  for children of currentVertex
    if children.visited == false
      dfsRecursive(children, visitedVertices)
```

# DFS Pseudocode

```
dfs(startVertex)
  for vertex in vertices
    vertex.visited = false
  dfsRecursive(startVertex, visitedVertices)
```

```
dfsRecursive(currentVertex, visitedVertices)
  currentVertex.visited = true
```

```
  for children of currentVertex
    if children.visited == false
      dfsRecursive(children, visitedVertices)
```

# DFS Pseudocode

```
dfs(startVertex)
  for vertex in vertices
    vertex.visited = false
  dfsRecursive(startVertex, visitedVertices)
```

```
dfsRecursive(currentVertex, visitedVertices)
  currentVertex.visited = true
```

```
  for children of currentVertex
    if children.visited == false
      dfsRecursive(children, visitedVertices)
```