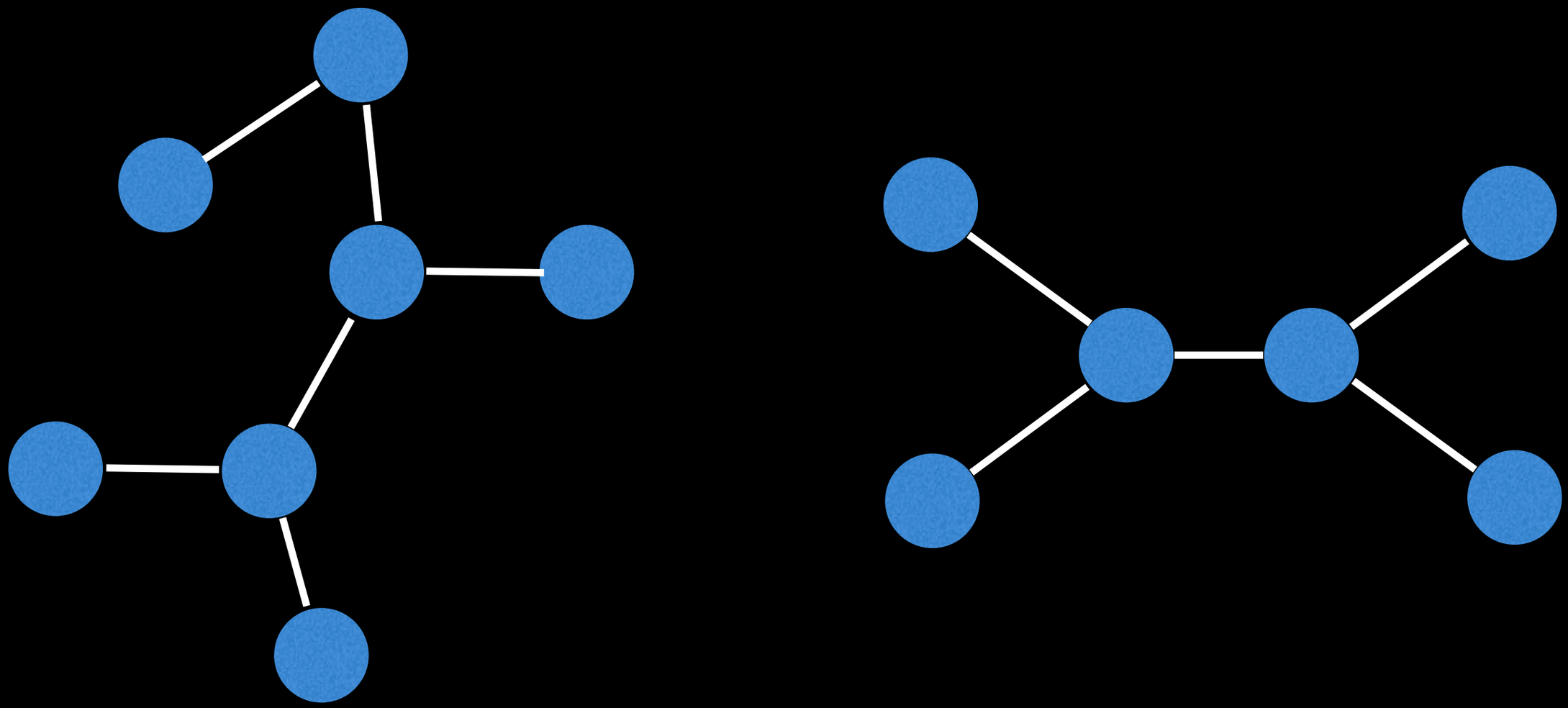
Center(s) of a tree

William Fiset

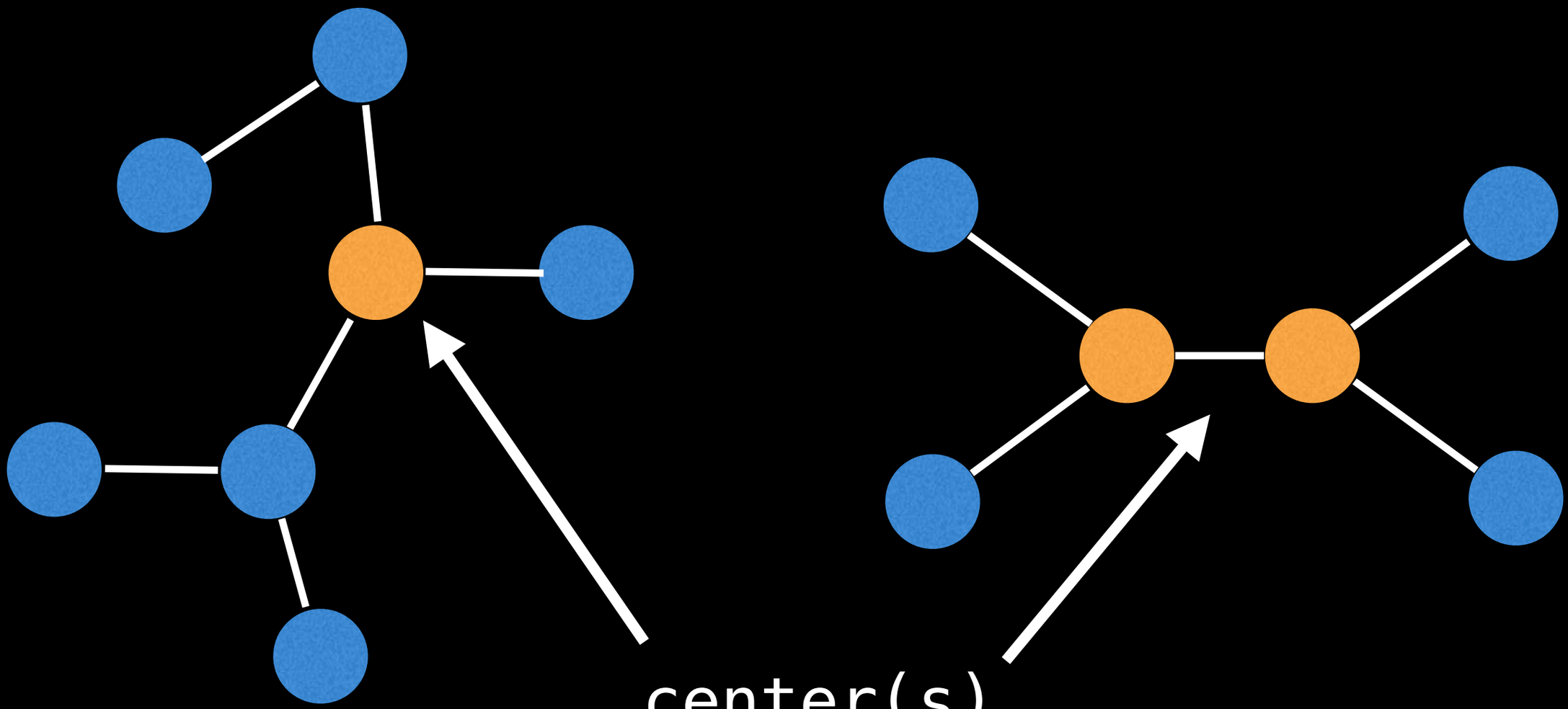
Center(s) of undirected tree

An interesting problem when you have an undirected tree is finding the tree's **center node(s)**. This could come in handy if we wanted to select a good node to root our tree 😉

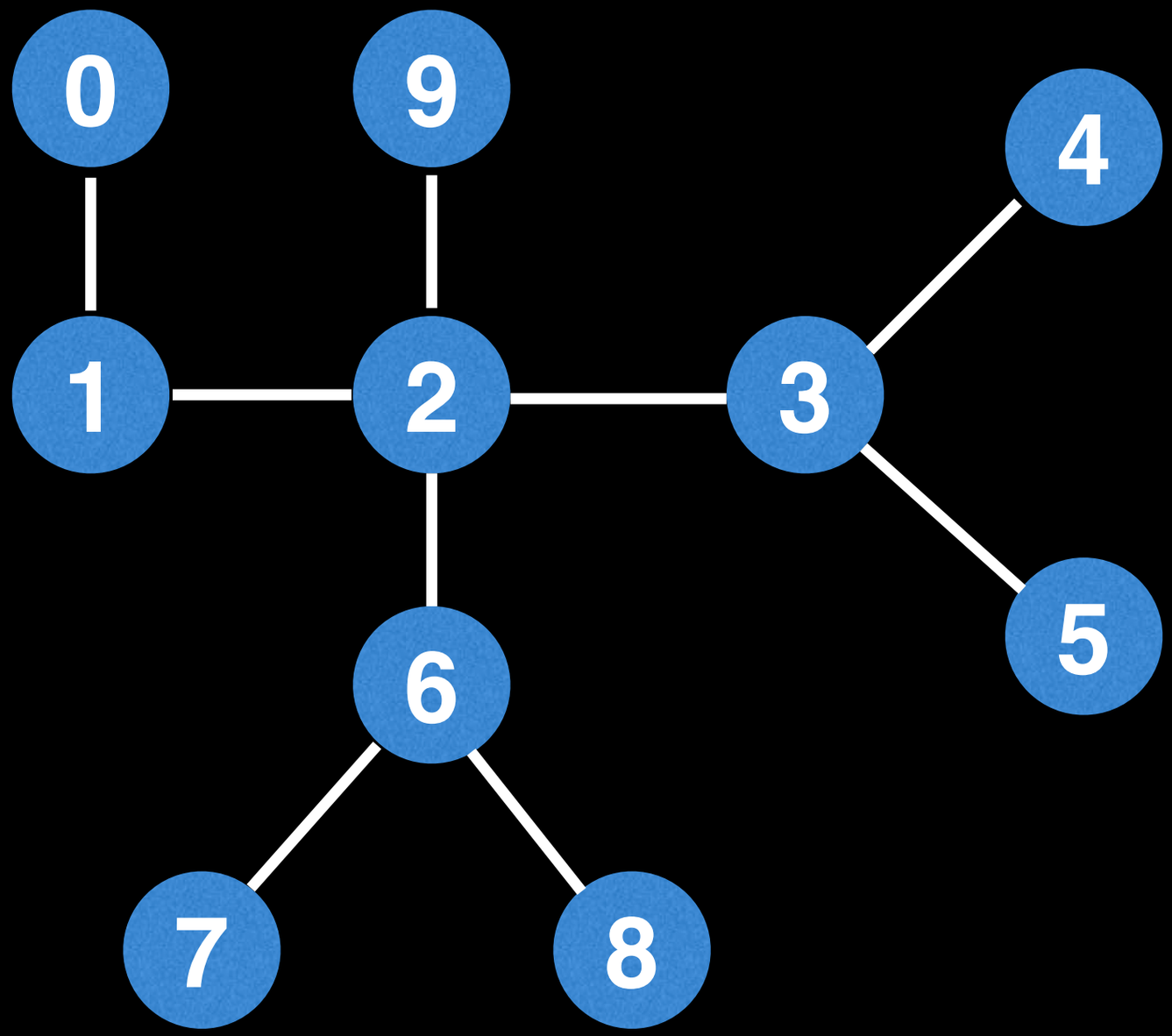


Center(s) of undirected tree

An interesting problem when you have an undirected tree is finding the tree's **center node(s)**. This could come in handy if we wanted to select a good node to root our tree 😉

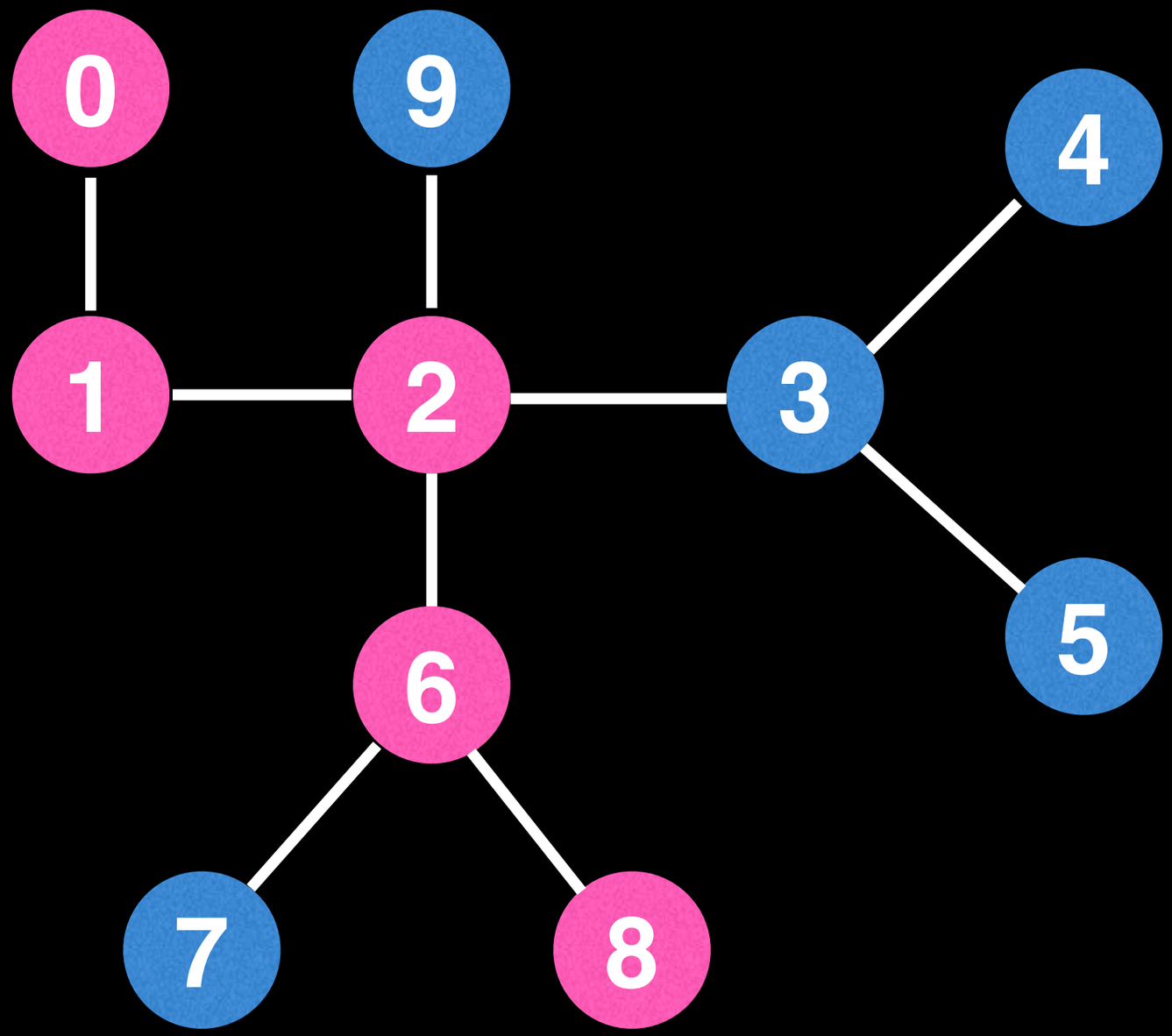


Center(s) of undirected tree



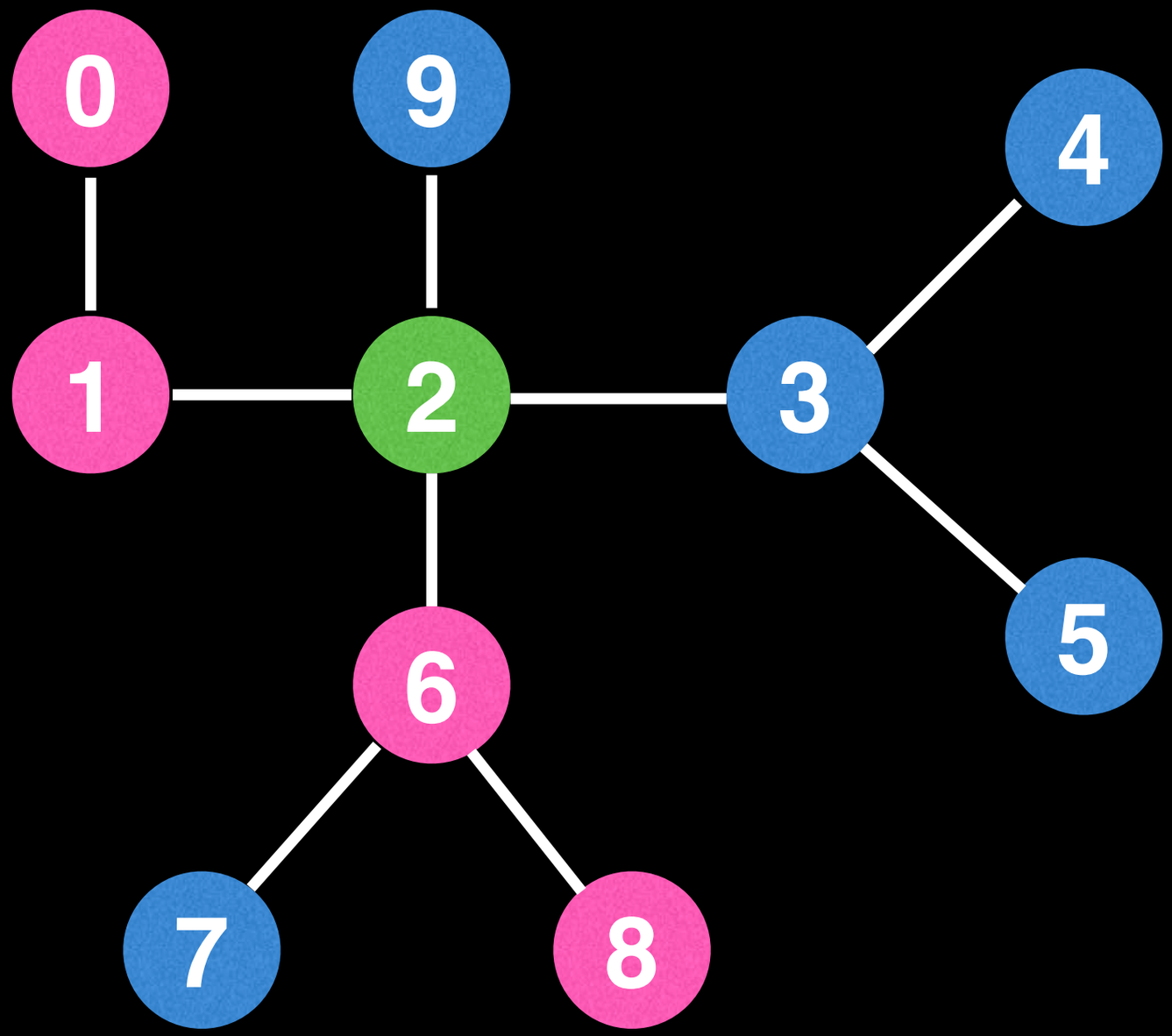
Notice that the center is always the middle vertex or middle two vertices in every longest path along the tree.

Center(s) of undirected tree



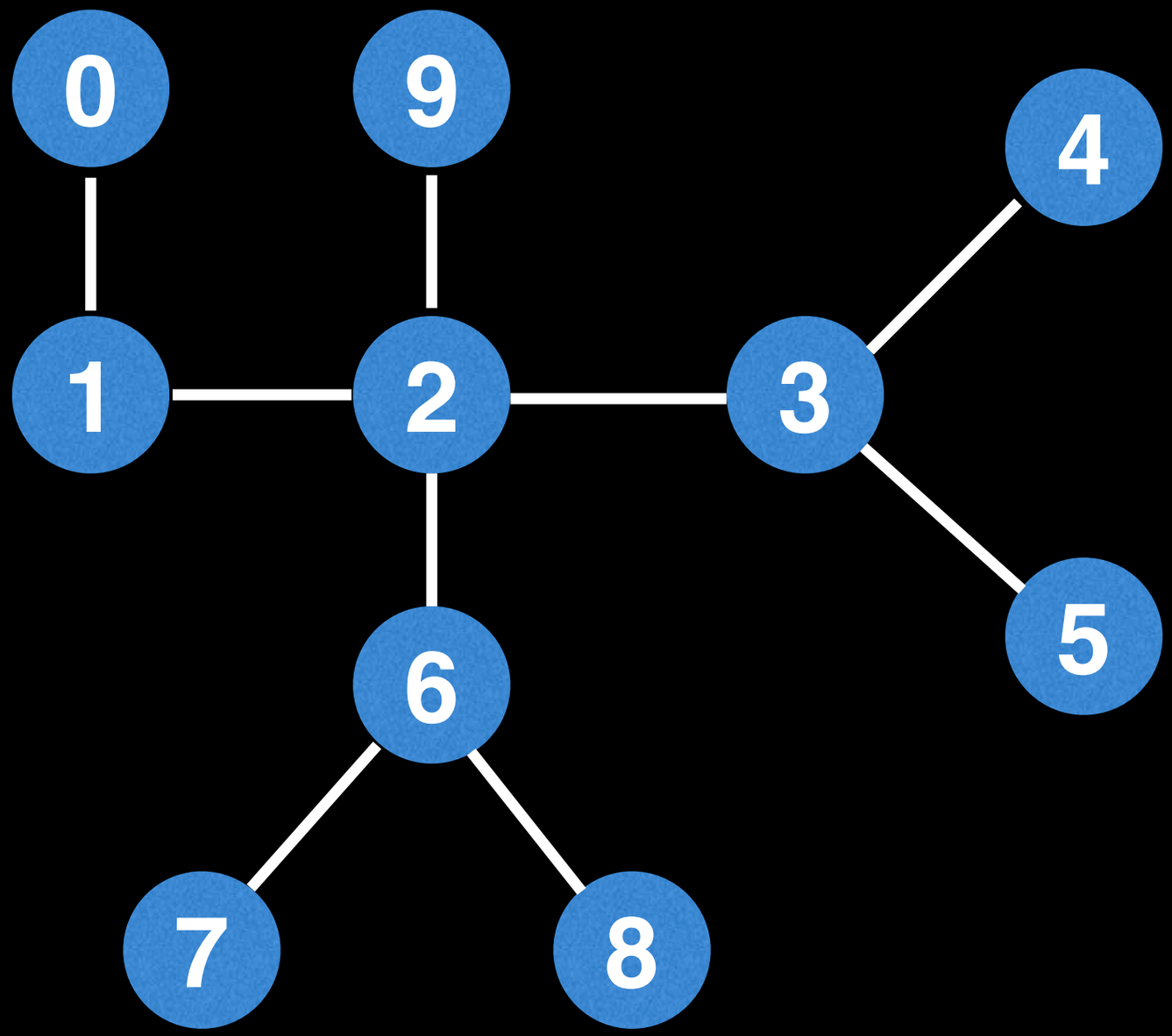
Notice that the center is always the middle vertex or middle two vertices in every longest path along the tree.

Center(s) of undirected tree



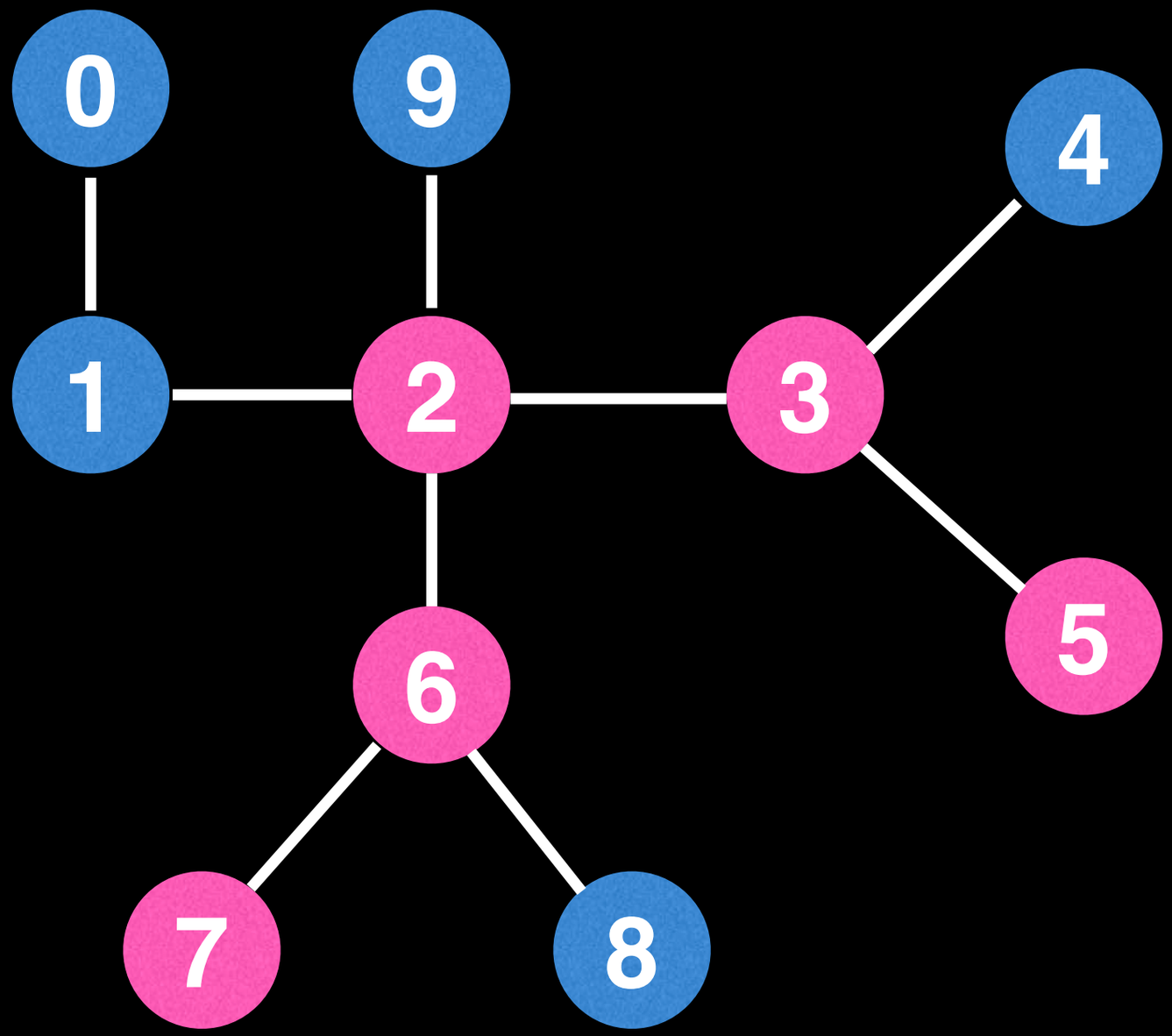
Notice that the center is always the middle vertex or middle two vertices in every longest path along the tree.

Center(s) of undirected tree



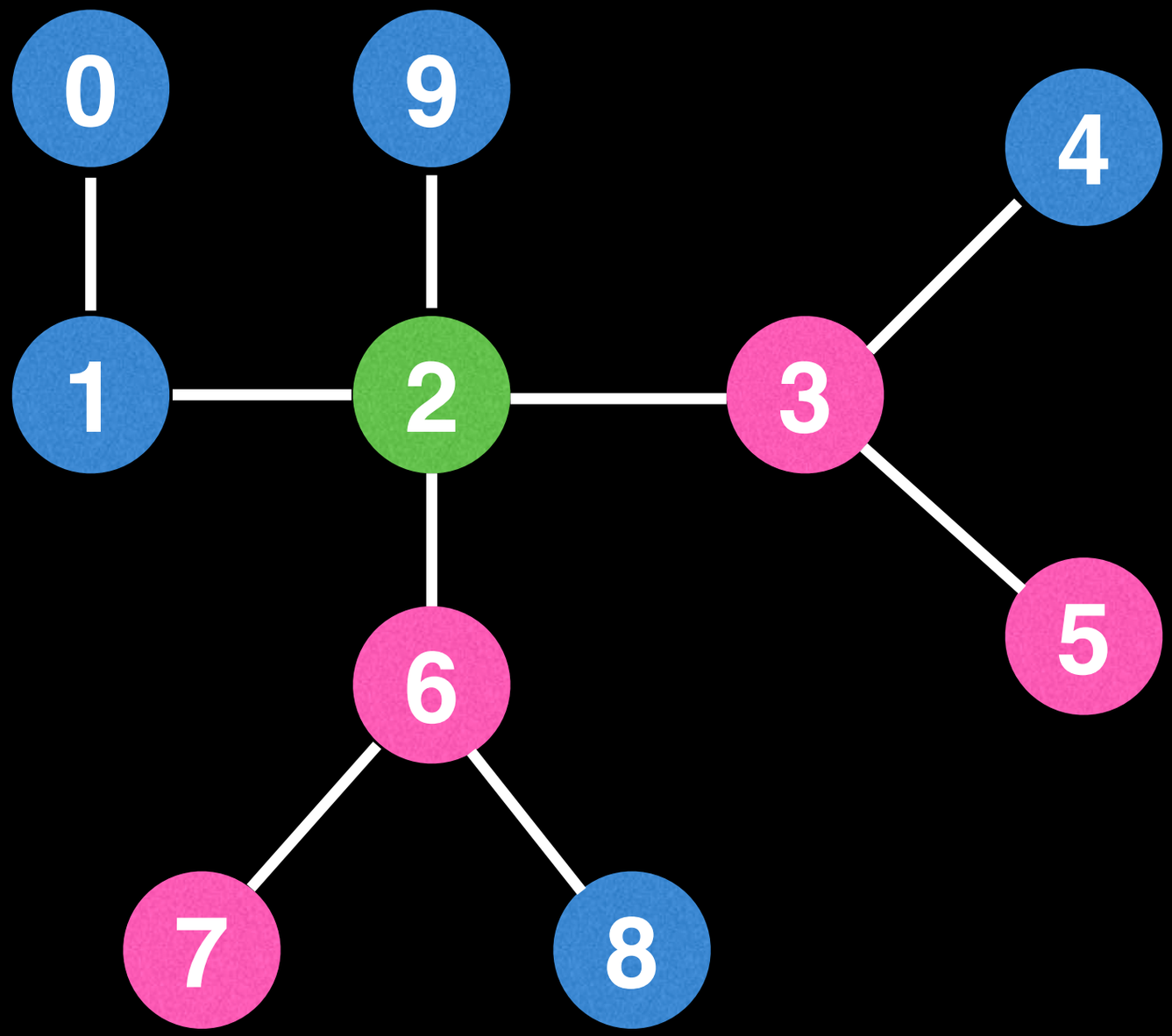
Notice that the center is always the middle vertex or middle two vertices in every longest path along the tree.

Center(s) of undirected tree



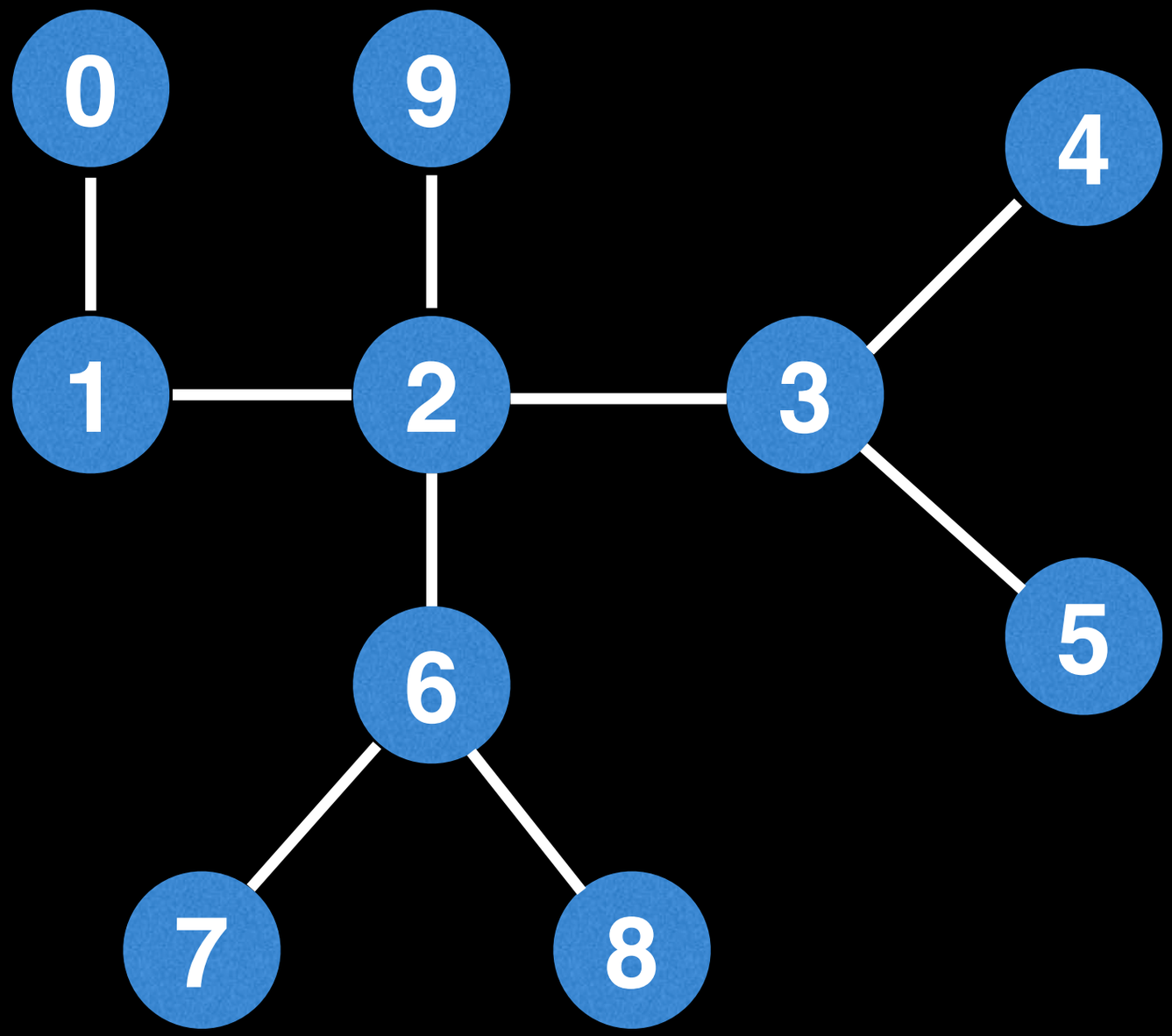
Notice that the center is always the middle vertex or middle two vertices in every longest path along the tree.

Center(s) of undirected tree



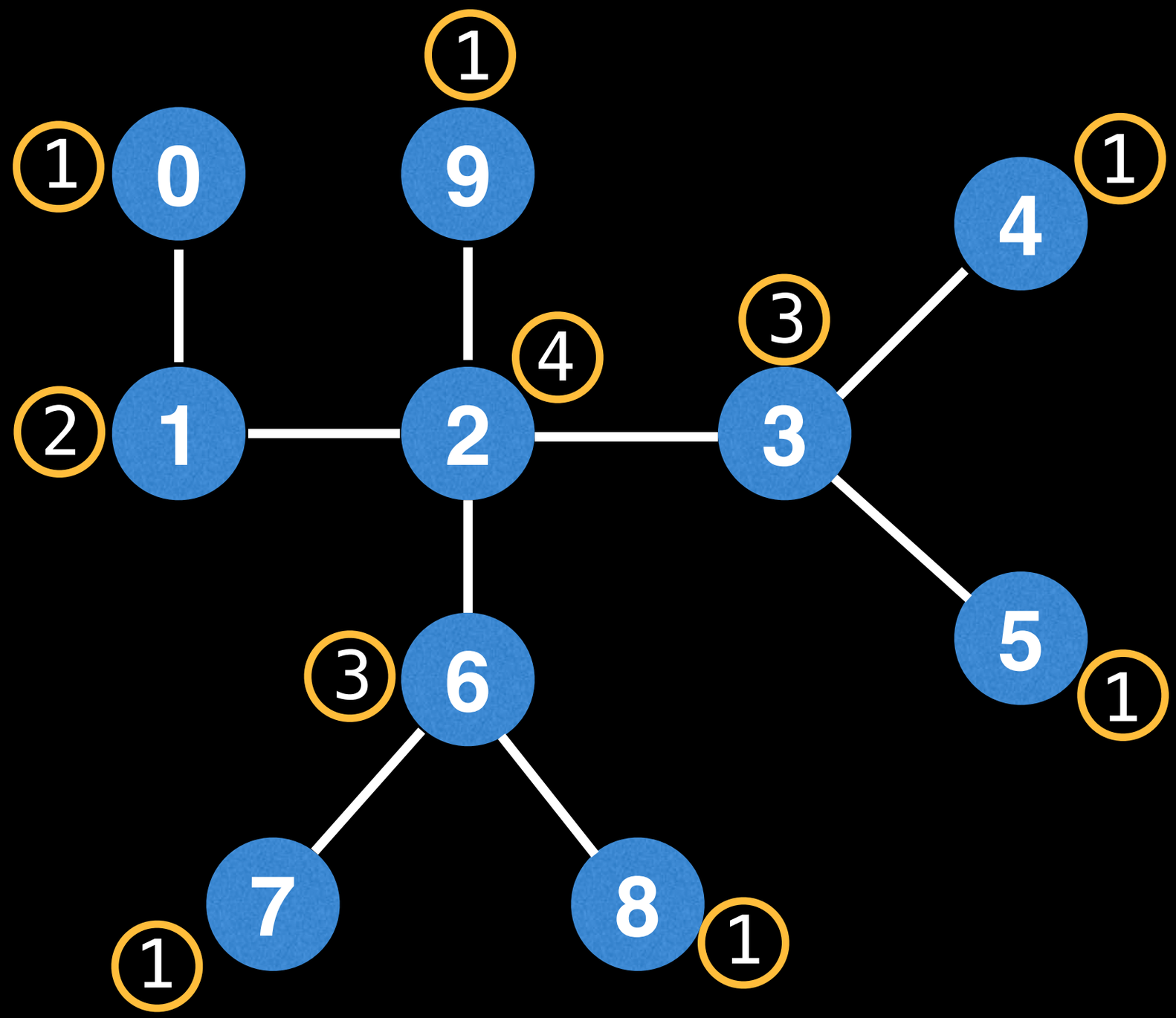
Notice that the center is always the middle vertex or middle two vertices in every longest path along the tree.

Center(s) of undirected tree



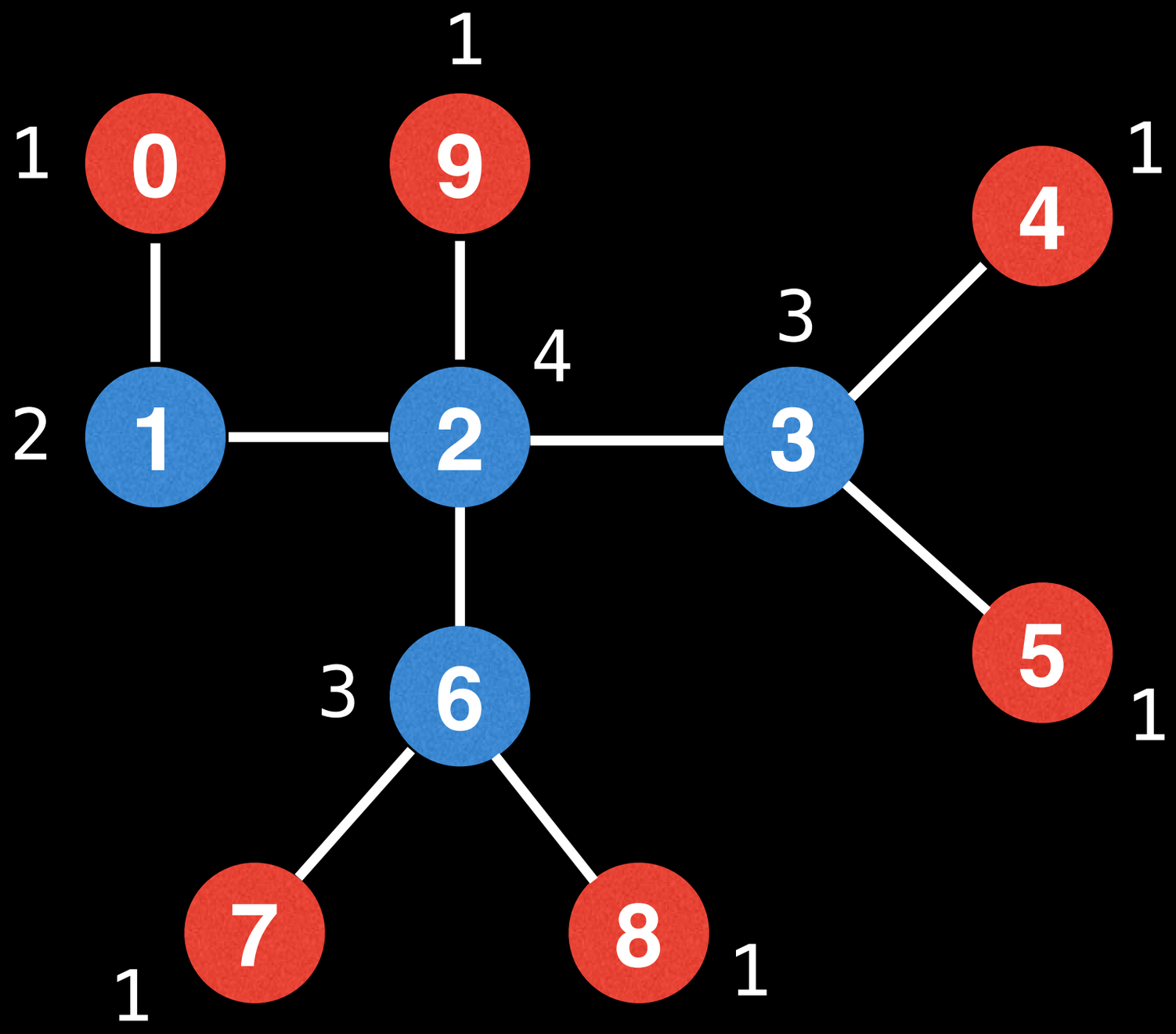
Another approach to find the center is to iteratively pick off each leaf node layer like we were peeling an onion.

Center(s) of undirected tree

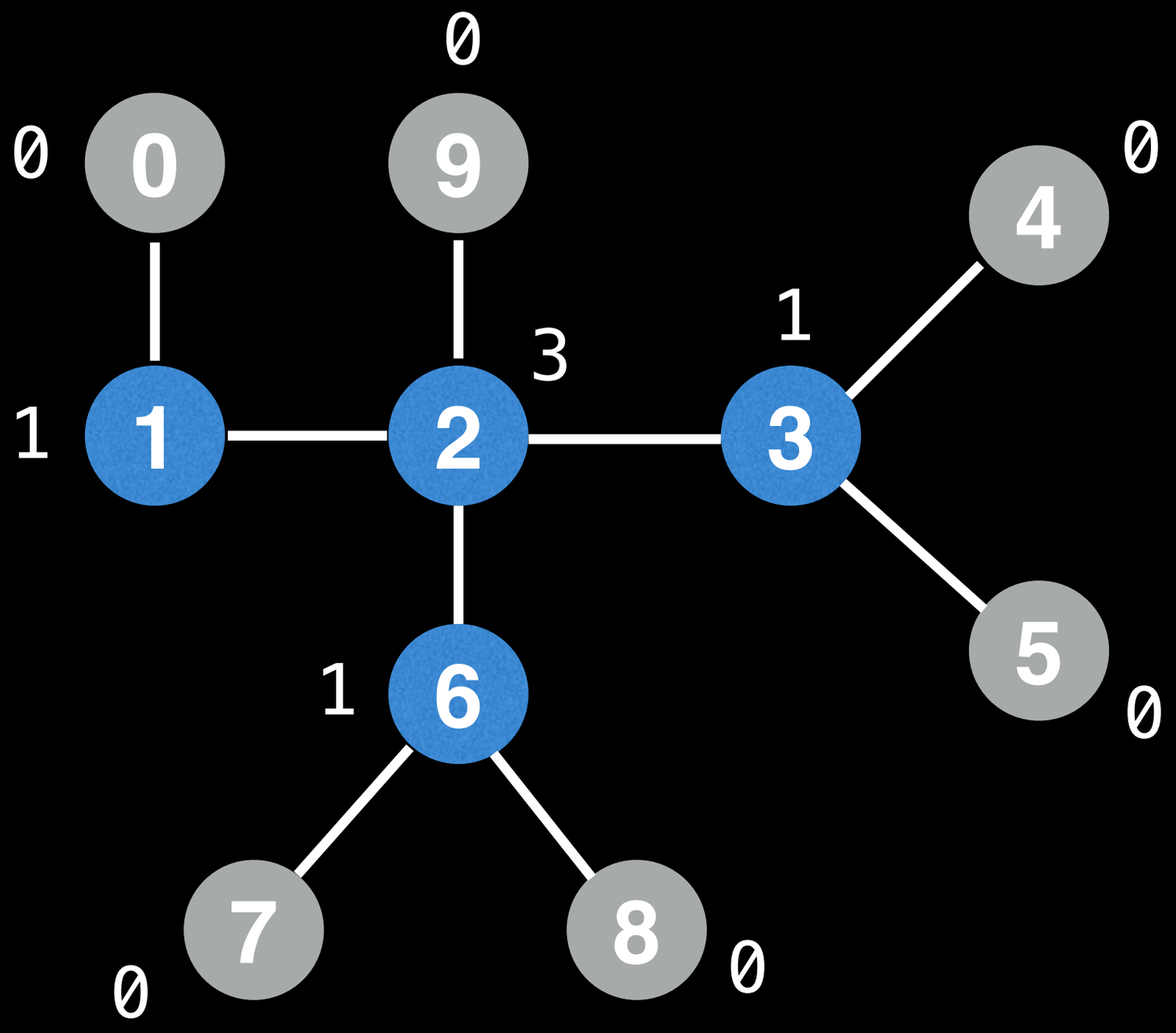


The orange circles represent the **degree** of each node. Observe that each leaf node will have a degree of 1.

Center(s) of undirected tree

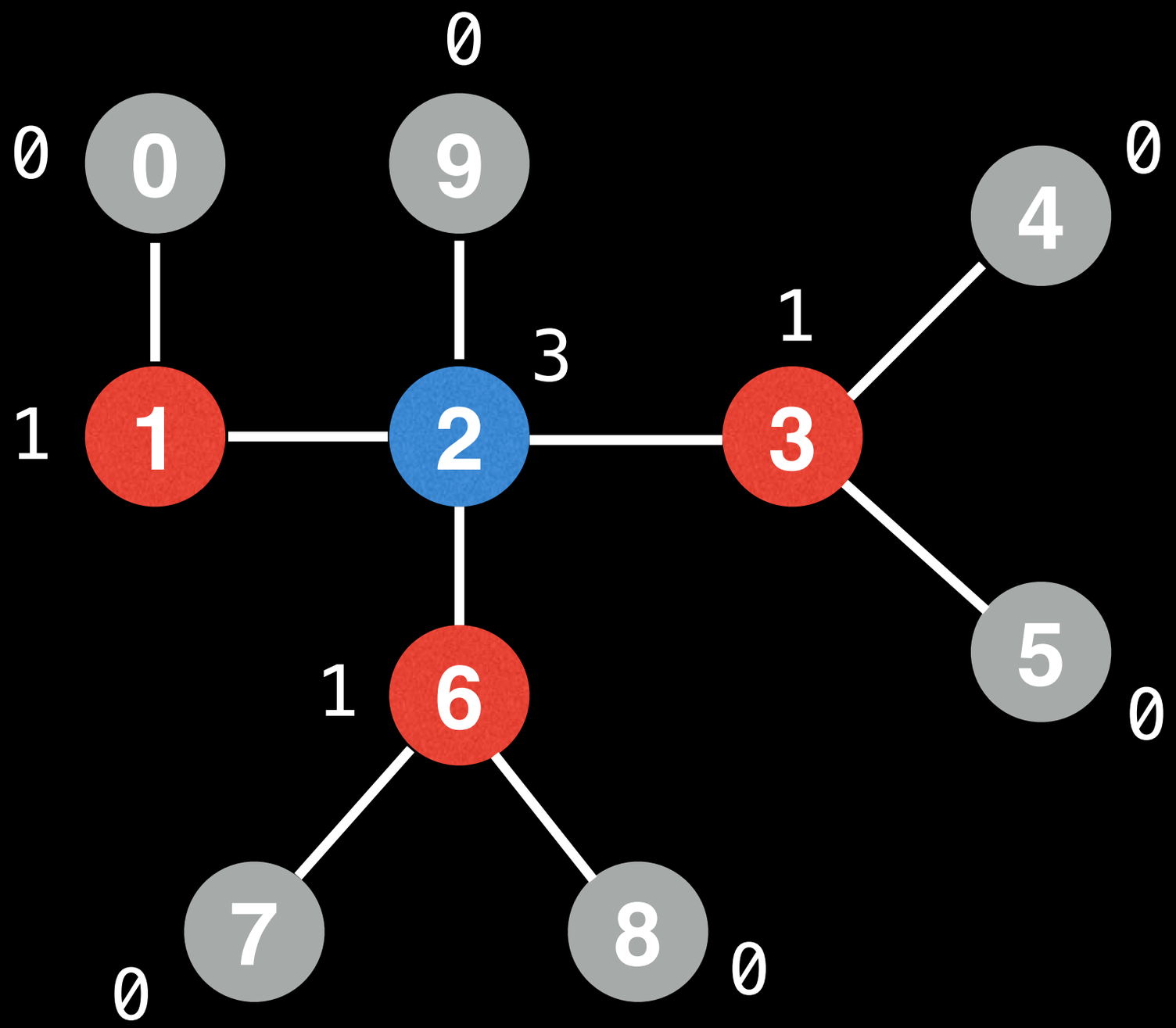


Center(s) of undirected tree

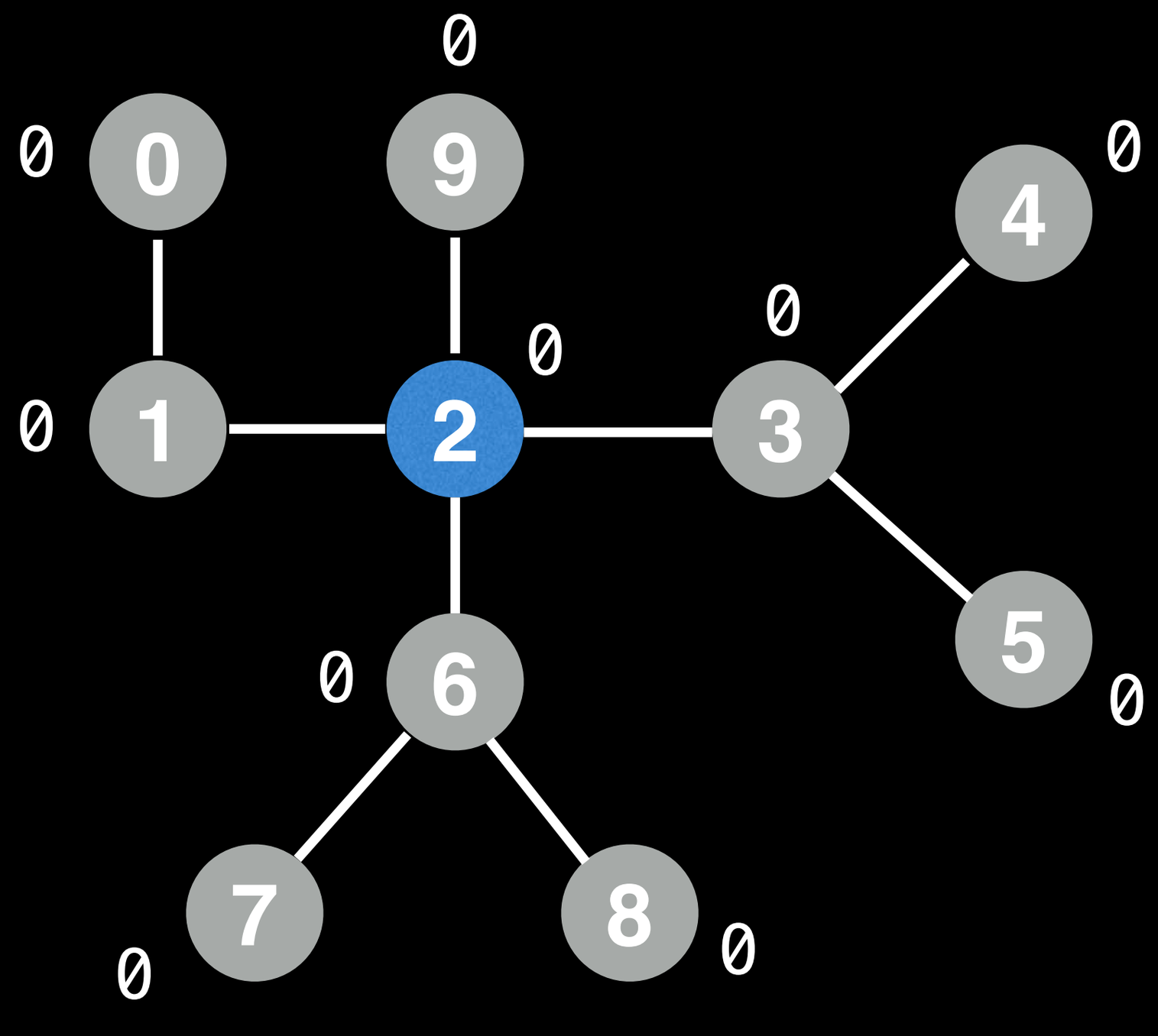


As we prune nodes also reduce the node degree values.

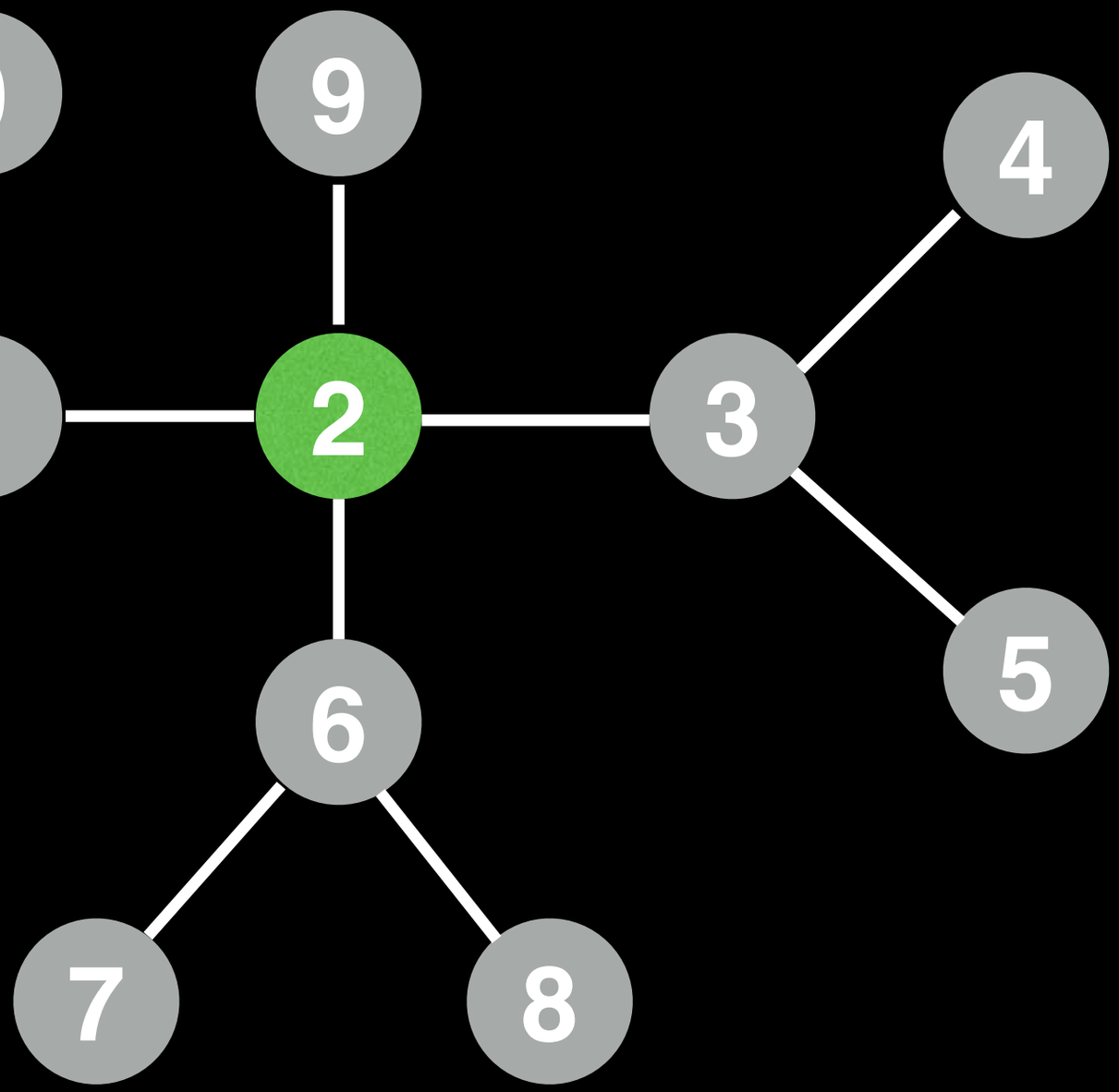
Center(s) of undirected tree



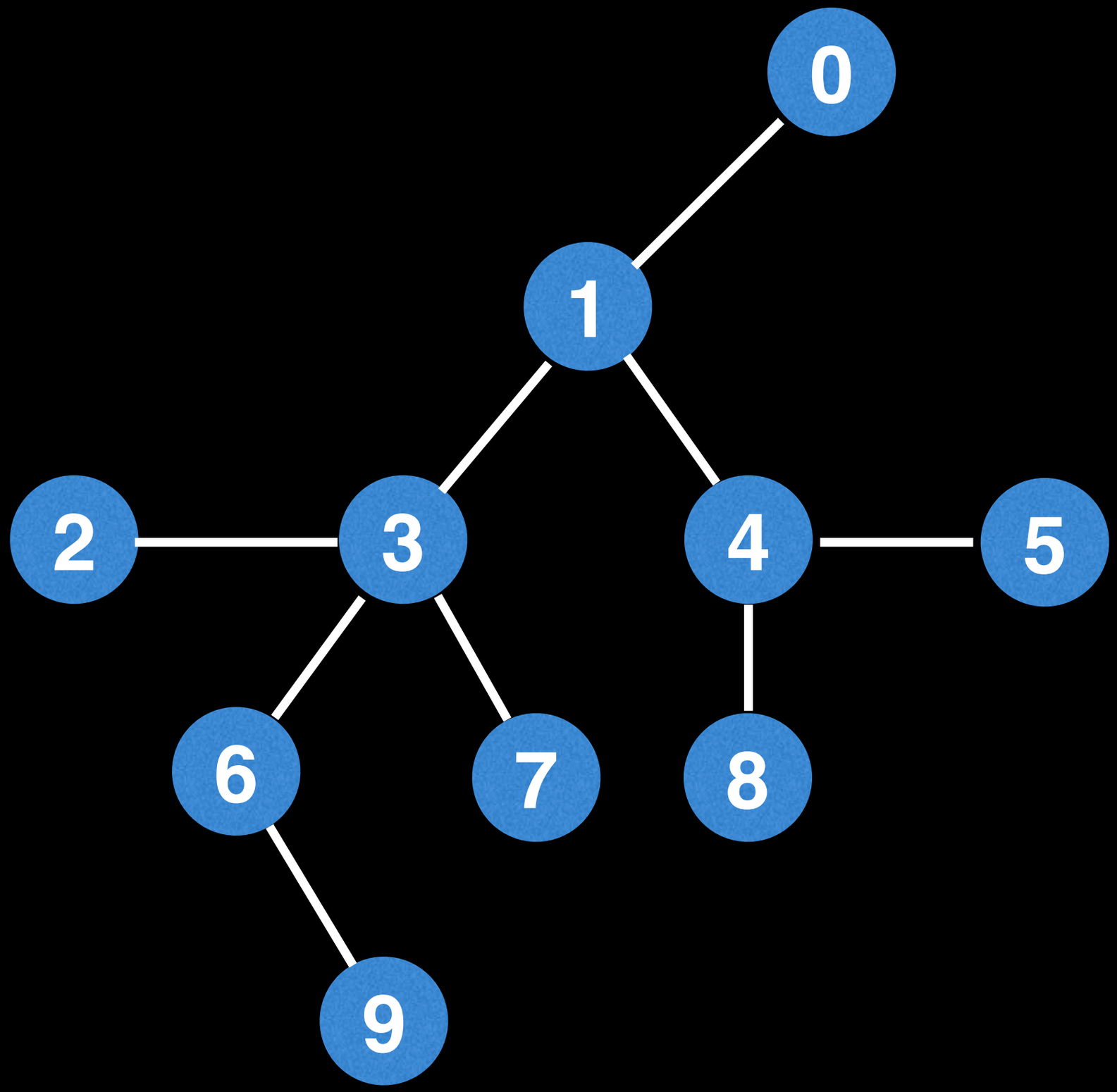
Center(s) of undirected tree



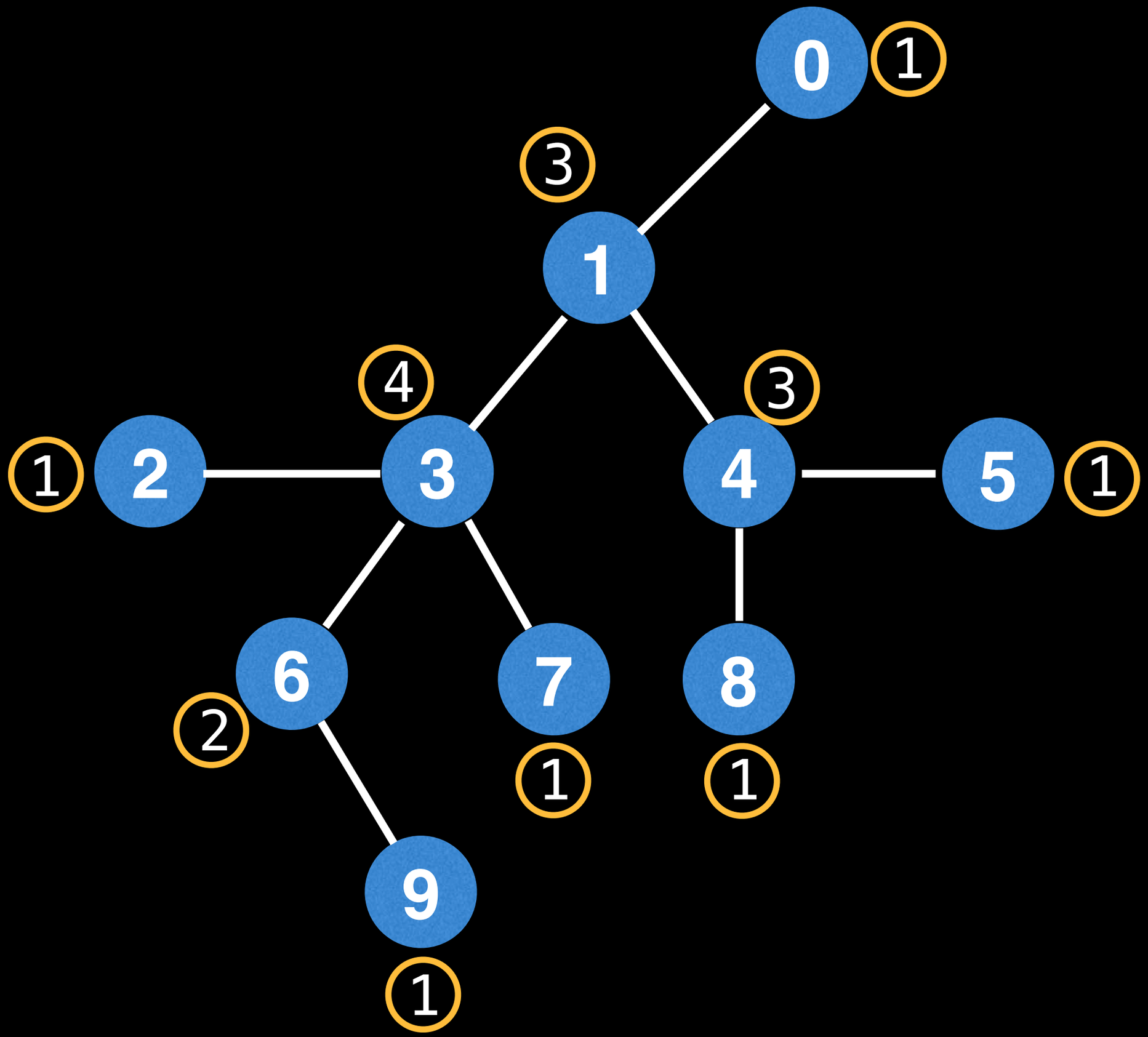
Center(s) of undirected tree



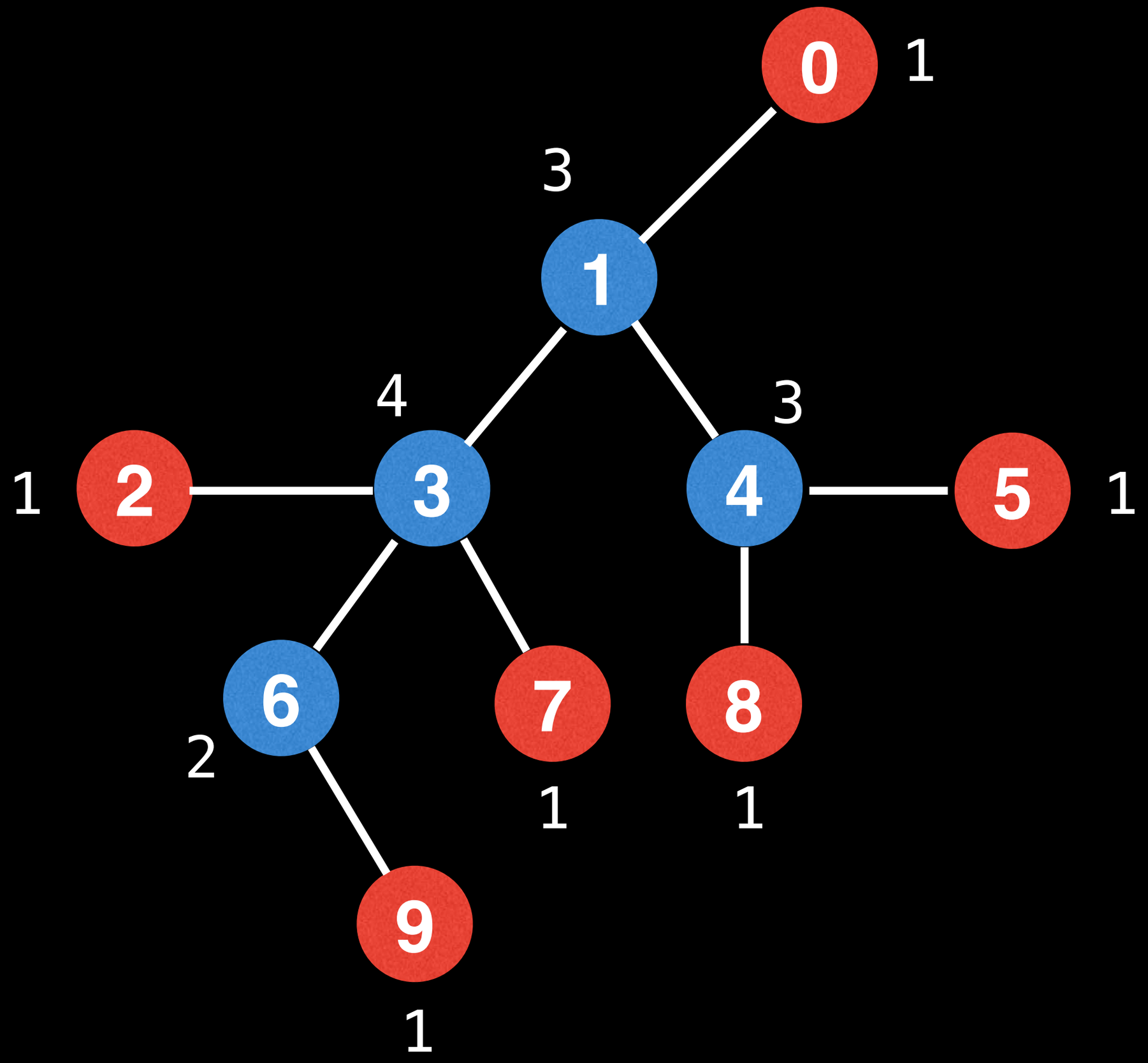
Center(s) of undirected tree



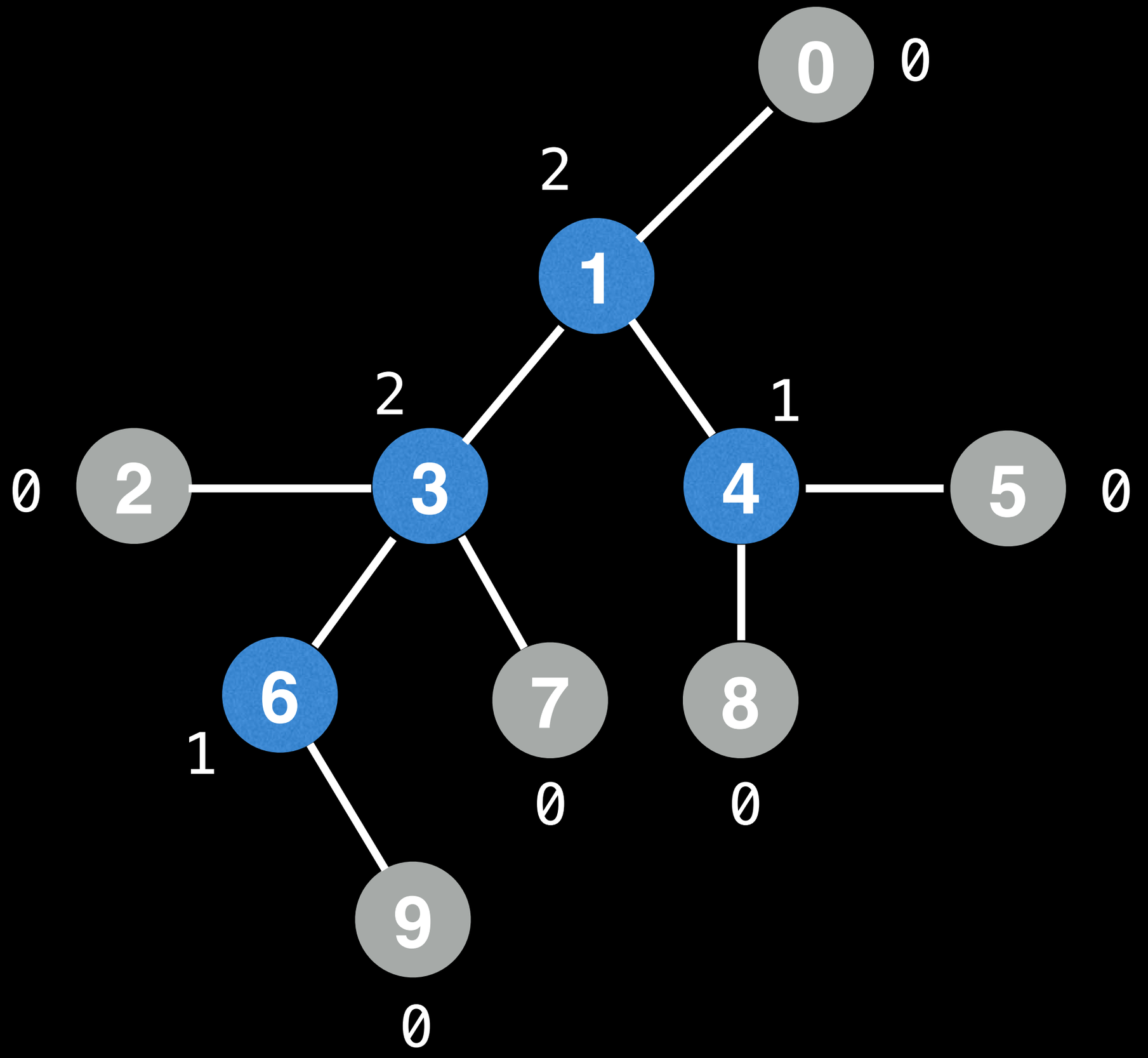
Center(s) of undirected tree



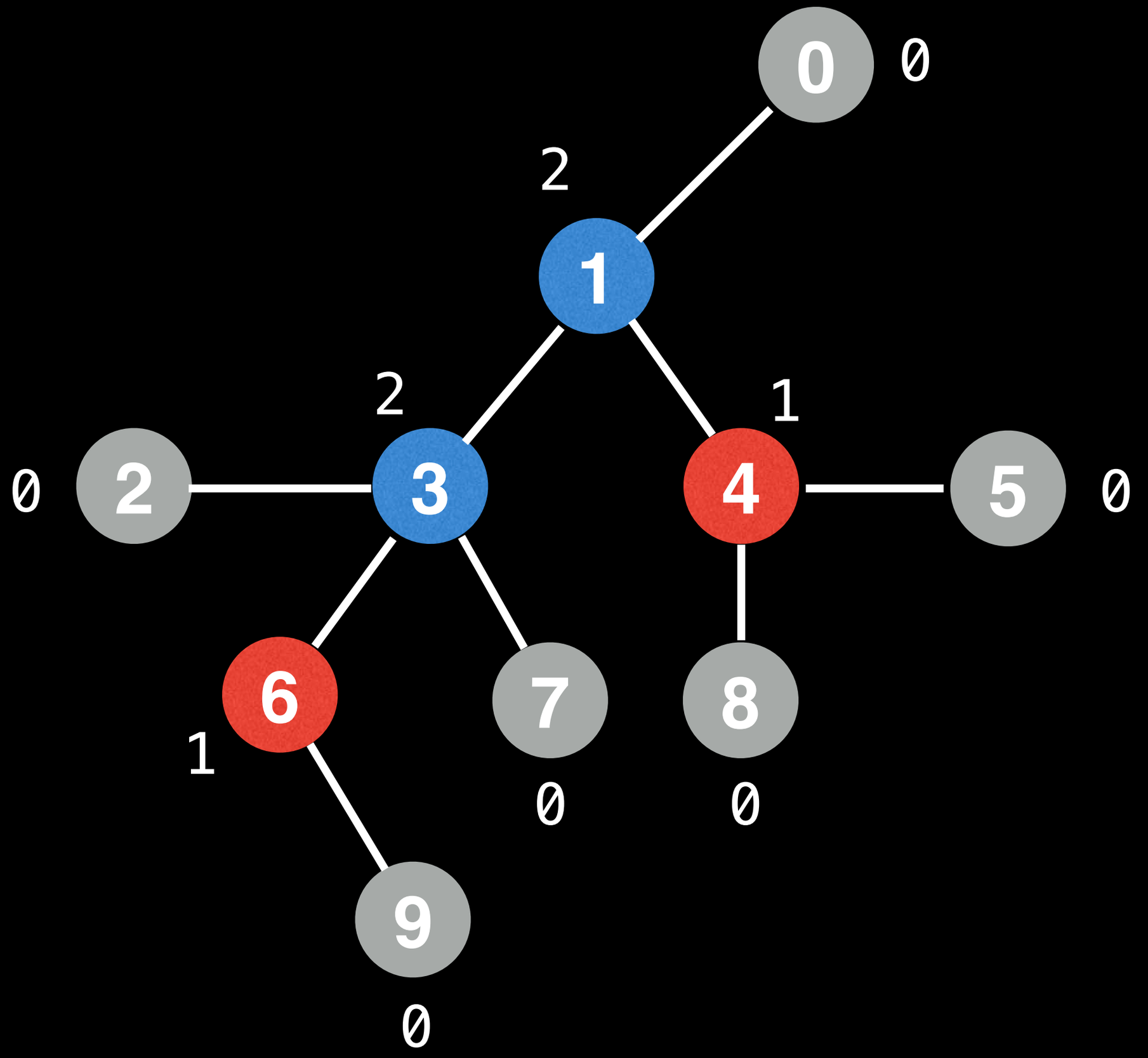
Center(s) of undirected tree



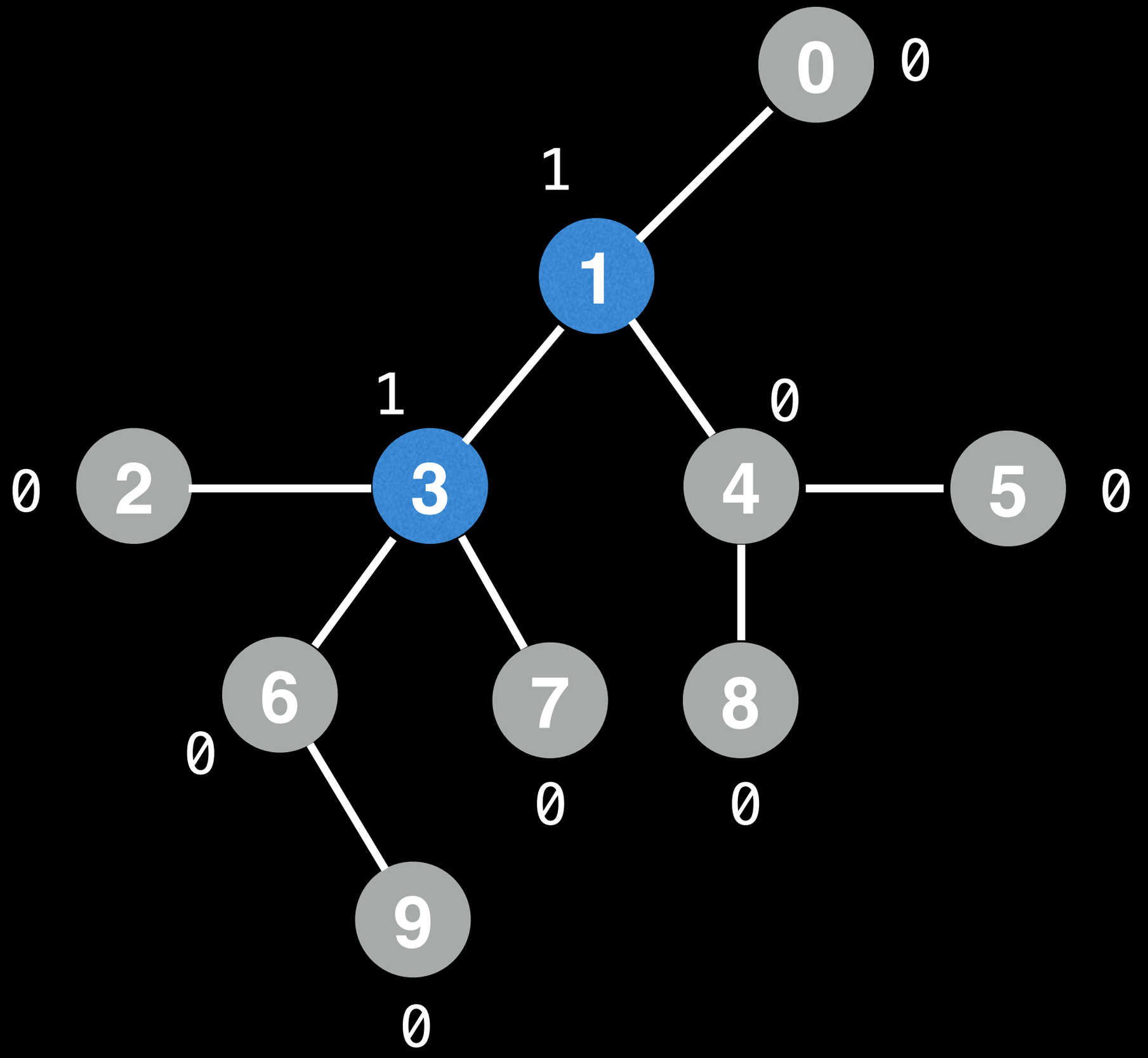
Center(s) of undirected tree



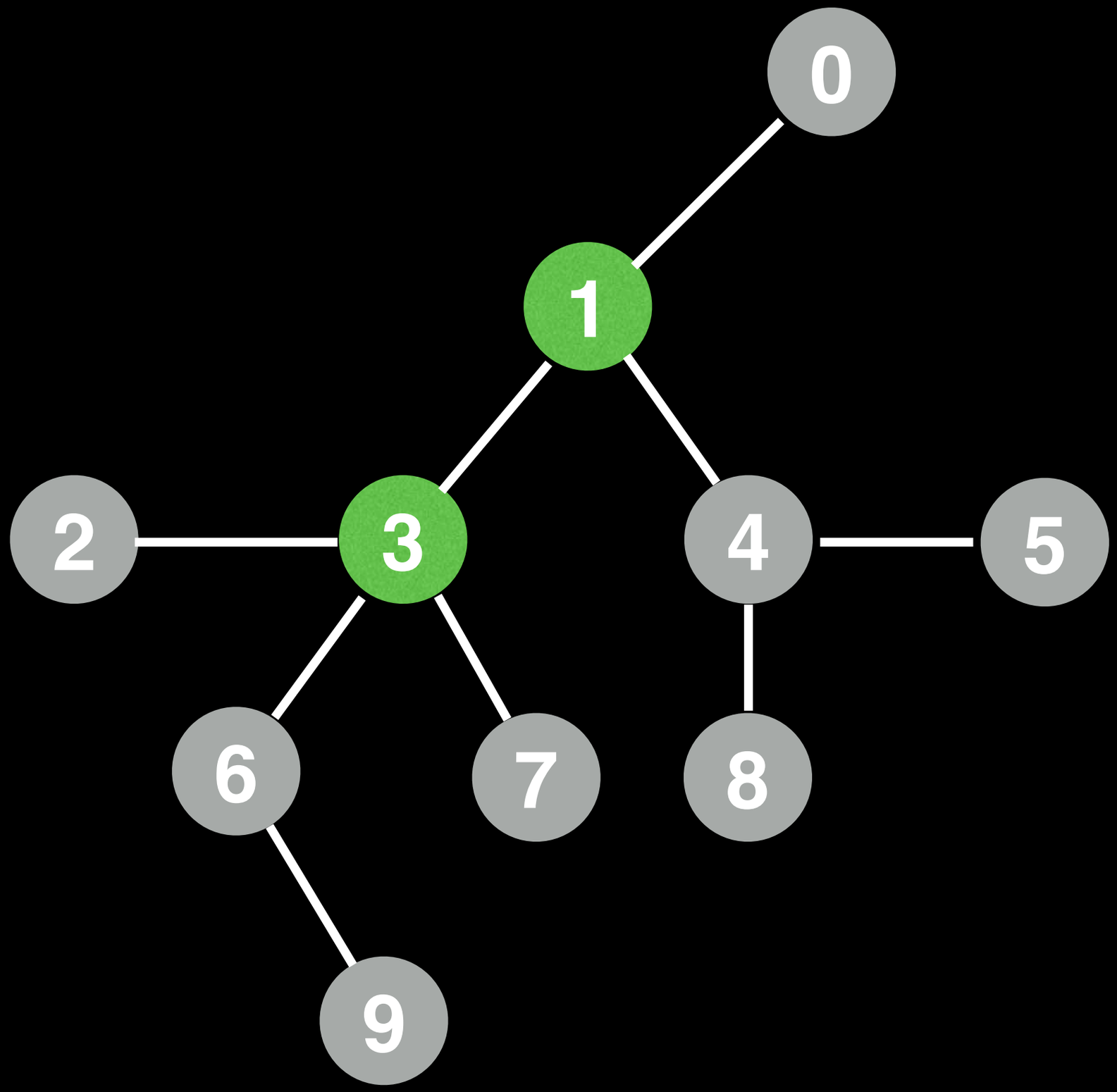
Center(s) of undirected tree



Center(s) of undirected tree



Center(s) of undirected tree



Some trees have two centers

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

```
# g = tree represented as an undirected graph
function treeCenters(g):
  n = g.numberOfNodes()
  degree = [0, 0, …, 0] # size n
  leaves = []
  for (i = 0; i < n; i++):
    degree[i] = g[i].size()
    if degree[i] == 0 or degree[i] == 1:
      leaves.add(i)
      degree[i] = 0
  count = leaves.size()
  while count < n:
    new_leaves = []
    for (node : leaves):
      for (neighbor : g[node]):
        degree[neighbor] = degree[neighbor] - 1
        if degree[neighbor] == 1:
          new_leaves.add(neighbor)
      degree[node] = 0
    count += new_leaves.size()
    leaves = new_leaves
  return leaves # center(s)
```

Center(s) of a Tree