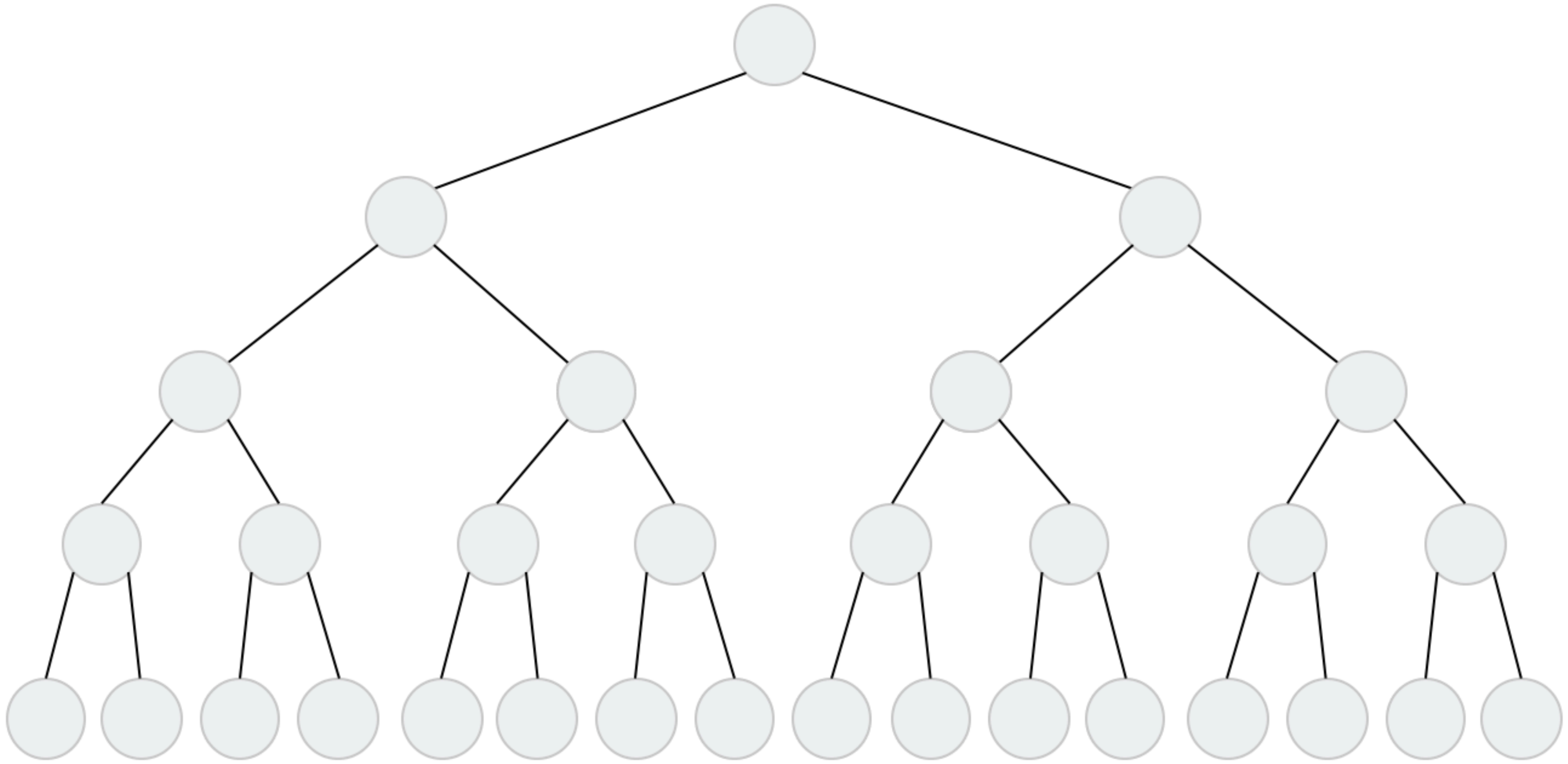
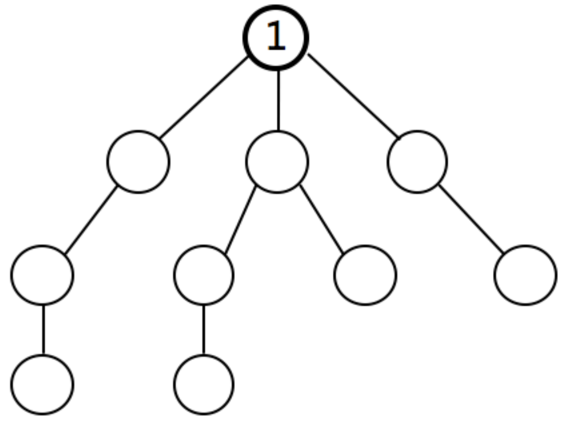
广度优先搜索(Breadth-First-Search)

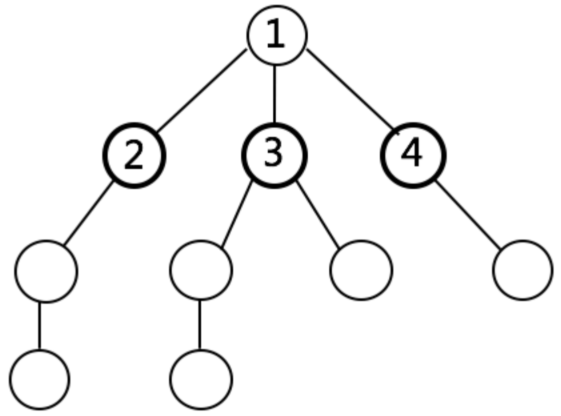
在树（图/状态集）中寻找特定节点



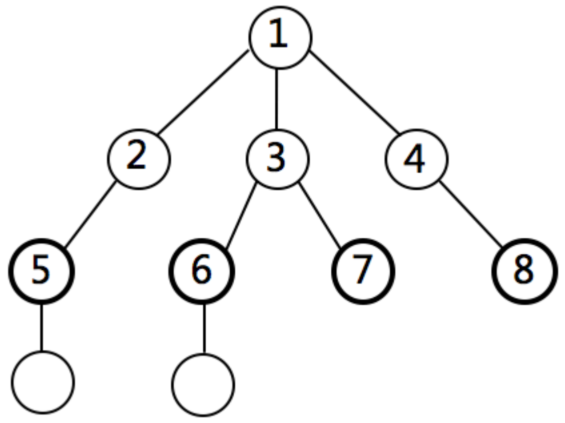
How the BFS would work



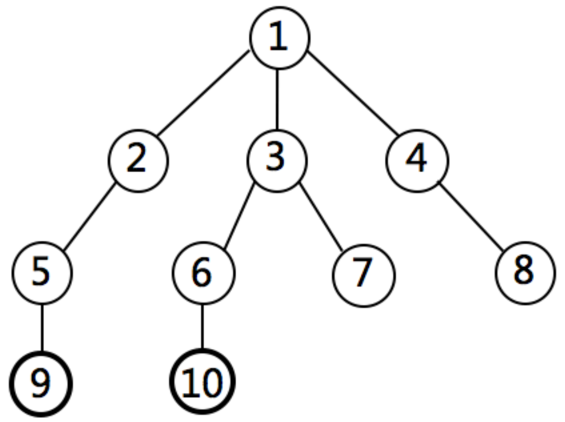
1



2



3



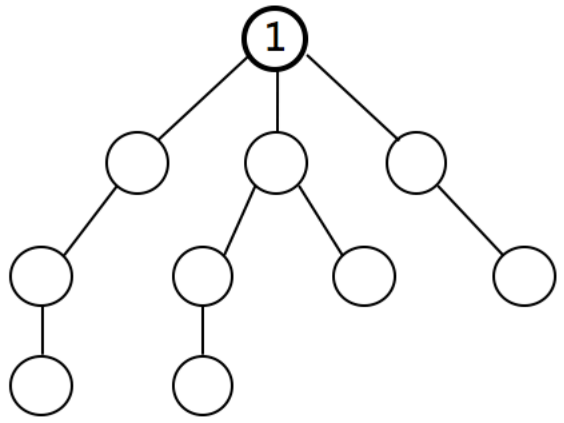
4

BFS代码

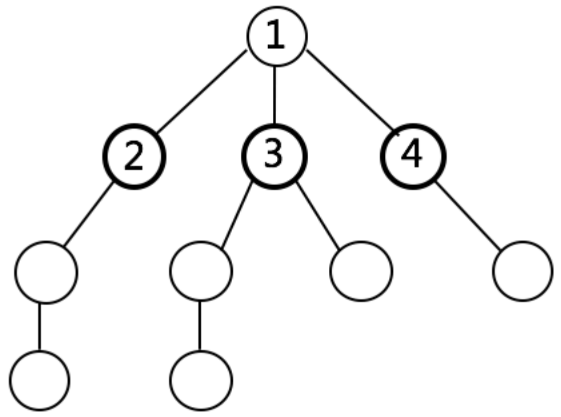
```
def BFS(graph, start, end):

    queue = []
    queue.append([start])
    visited.add(start)

    while queue:
        node = queue.pop()
        visited.add(node)

        process(node)
        nodes = generate_related_nodes(node)
        queue.push(nodes)

    # other processing work
    ...
```

深度优先搜索(Depth-First-Search)

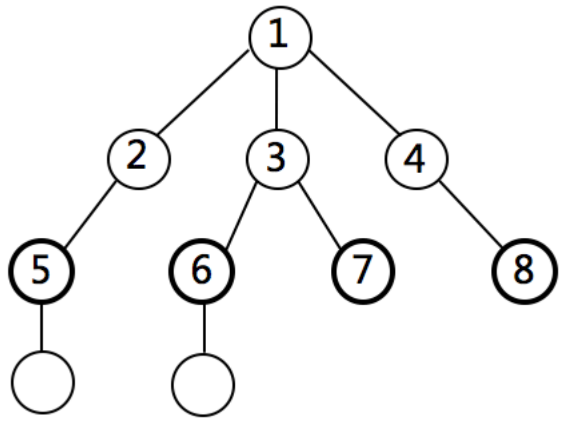
How a BFS Would Traverse This Tree



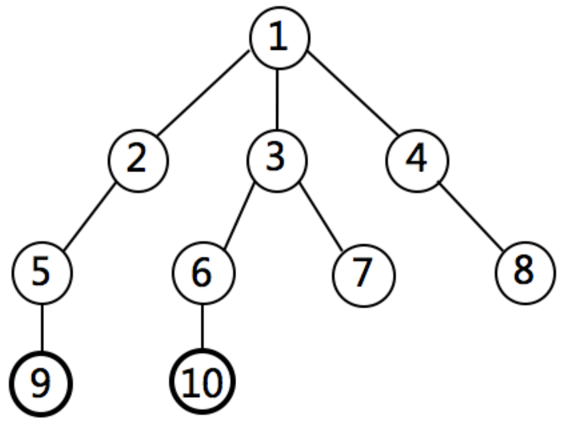
1



2

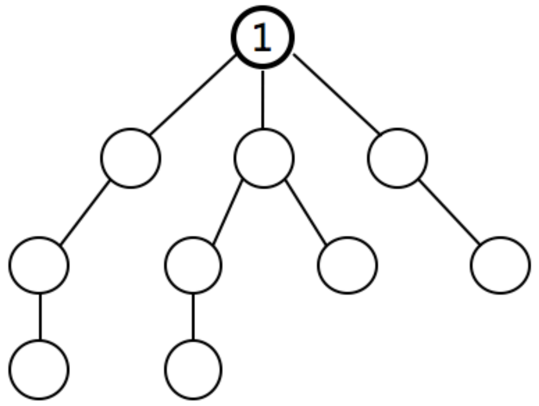


3

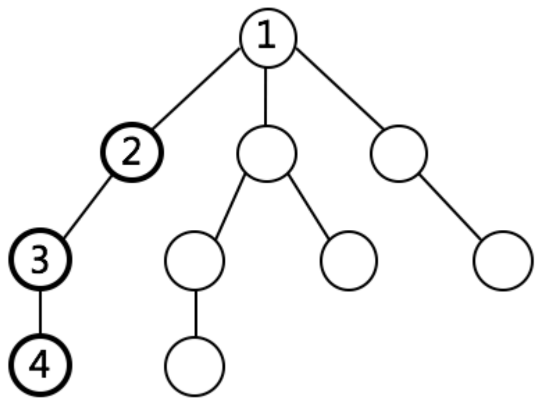


4

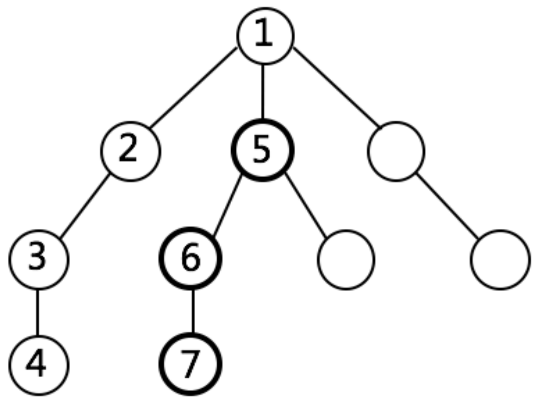
How a DFS Would Traverse This Tree



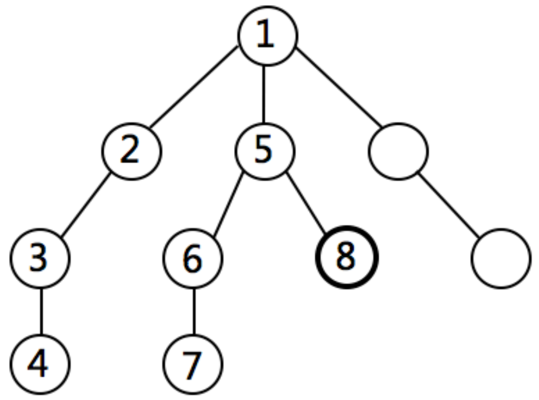
1



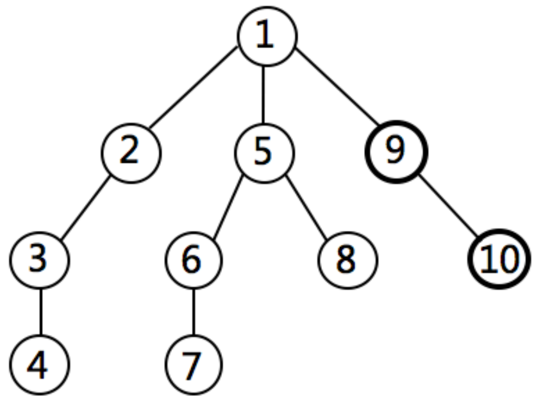
2



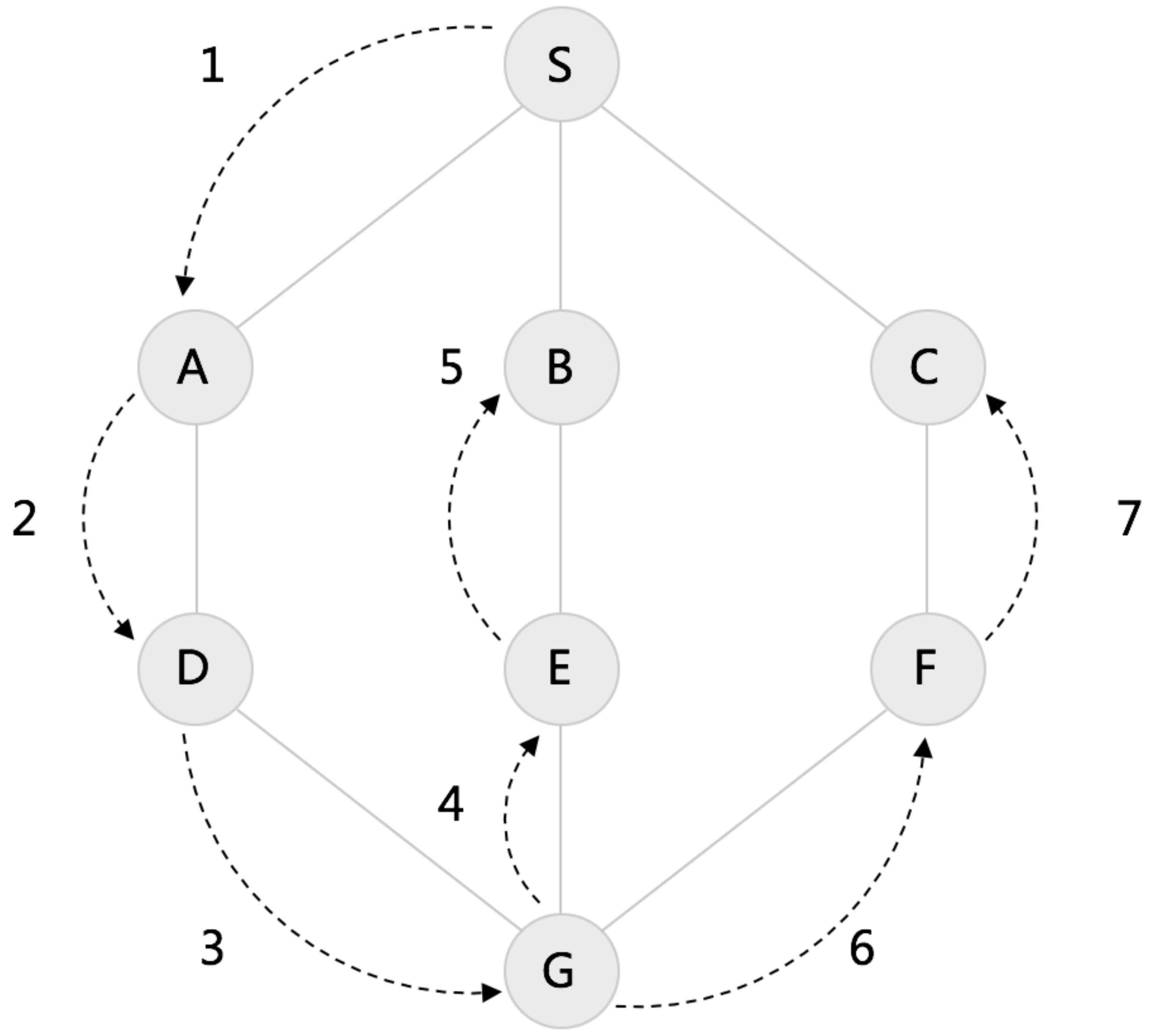
3

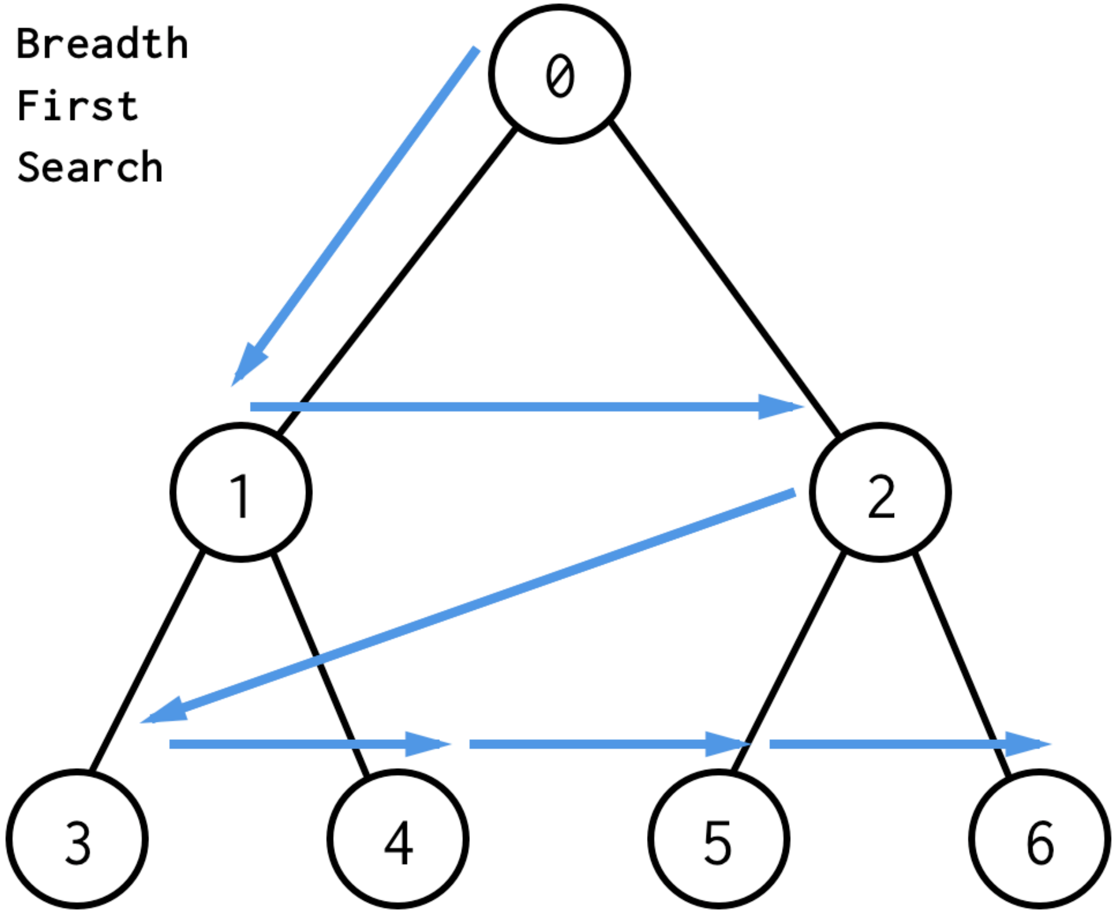


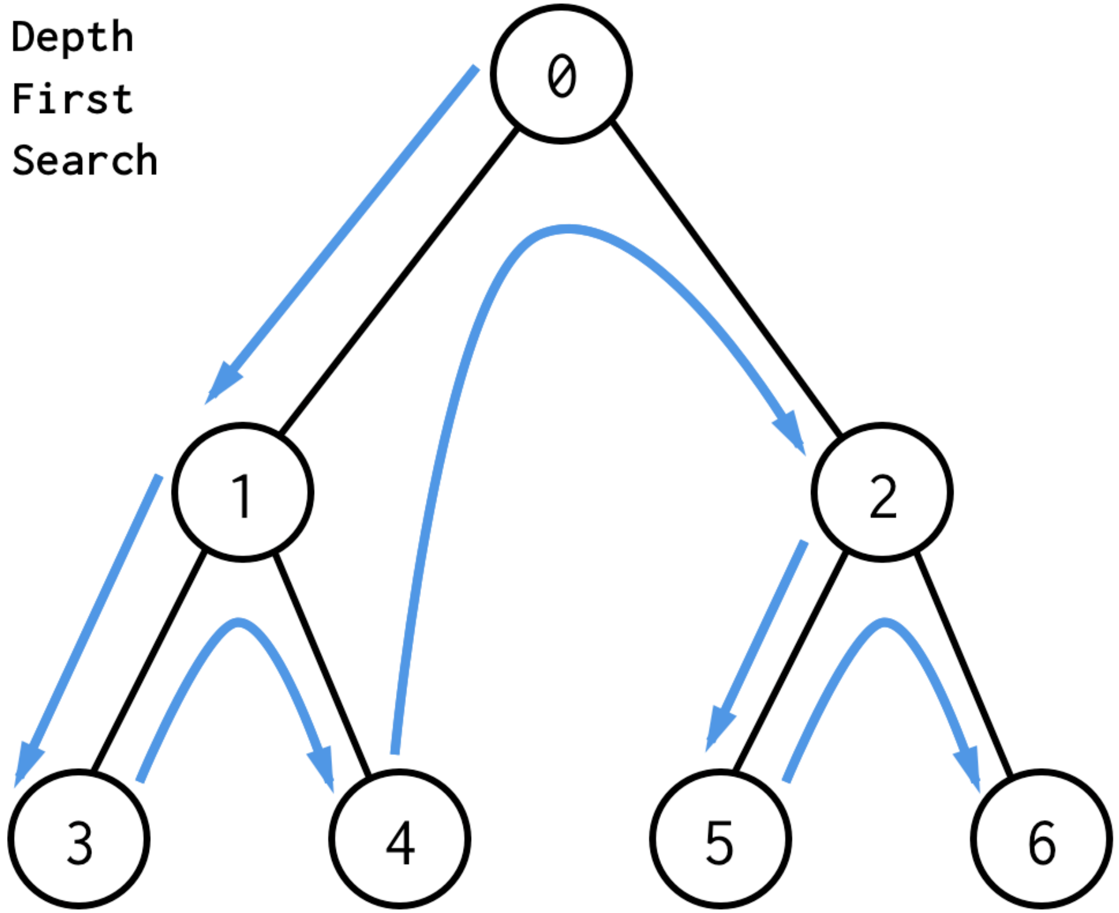
4



5


S
A
B
C
D
E
F
G
1
2
3
4
5
6
7


Breadth
First
Search
0
1
2
3
4
5
6


Depth
First
Search
0
1
2
3
4
5
6

DFS代码 - 递归写法

```python
visited = set()
def dfs(node, visited):
    visited.add(node)
    # process current node here.
    ...
    for next_node in node.children():
        if not next_node in visited:
            dfs(next_node, visited)
```

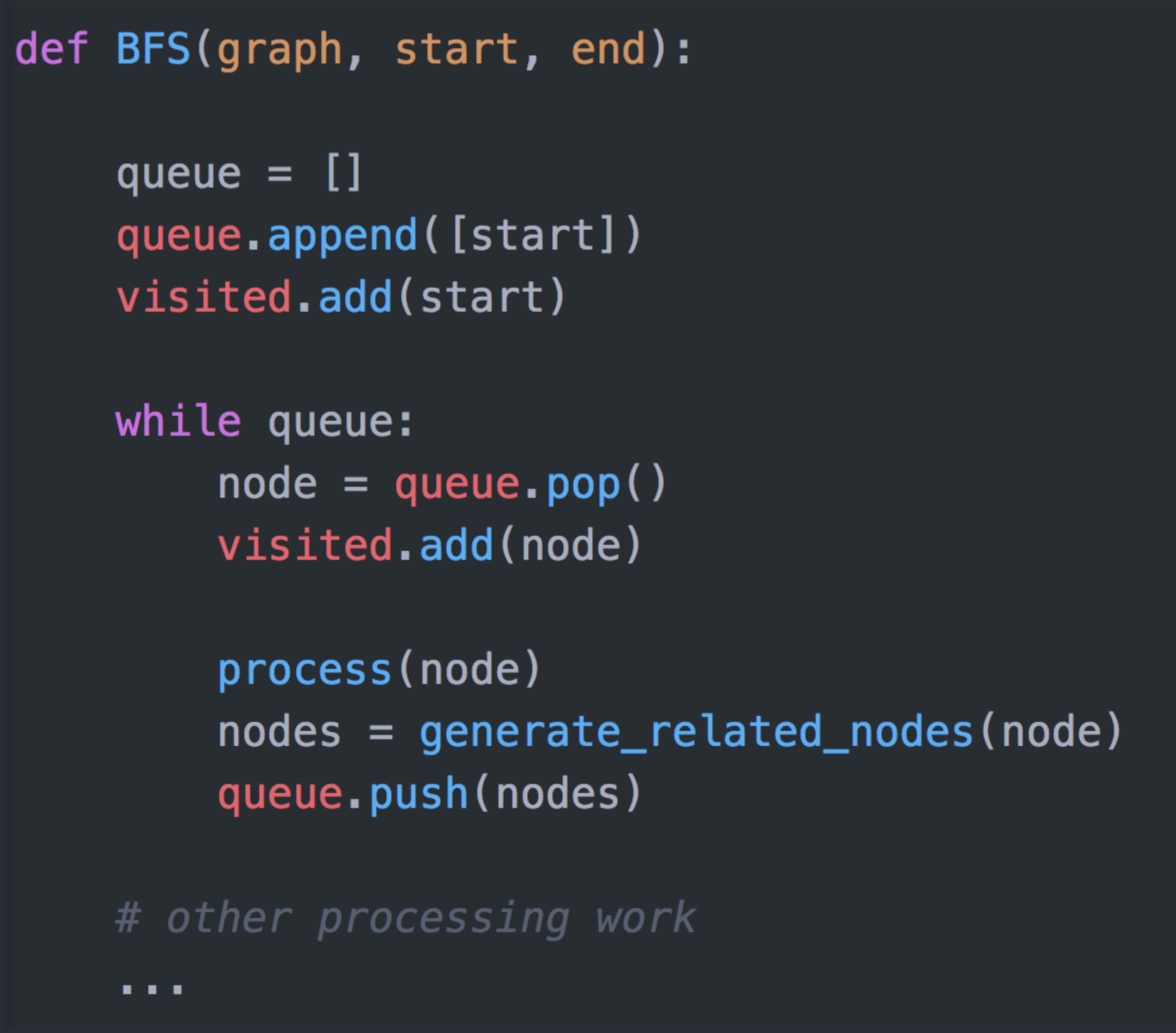
DFS代码 - 非递归写法

```python
def DFS(self, tree):

    if tree.root is None:
        return []

    visited, stack = [], [tree.root]

    while stack:
        node = stack.pop()
        visited.add(node)

        process(node)
        nodes = generate_related_nodes(node)
        stack.push(nodes)

    # other processing work
    ...
```

DFS代码 - 递归写法

```python
visited = set()
def dfs(node, visited):
    visited.add(node)
    # process current node here.
    ...
    for next_node in node.children():
        if not next_node in visited:
            dfs(next_node, visited)
```

BFS代码



```
def BFS(graph, start, end):

    queue = []
    queue.append([start])
    visited.add(start)

    while queue:
        node = queue.pop()
        visited.add(node)

        process(node)
        nodes = generate_related_nodes(node)
        queue.push(nodes)

    # other processing work
    ...
```