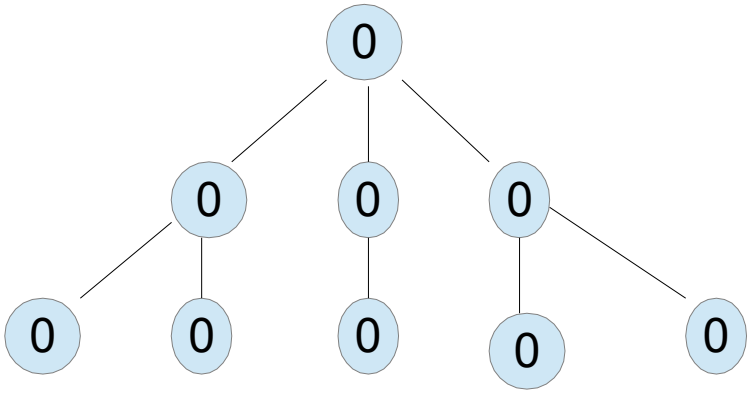
Basic Depth First

```
DepthFirst(Graph G with set vertices V & edges E):
    mark all vertices with 0
    count =0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

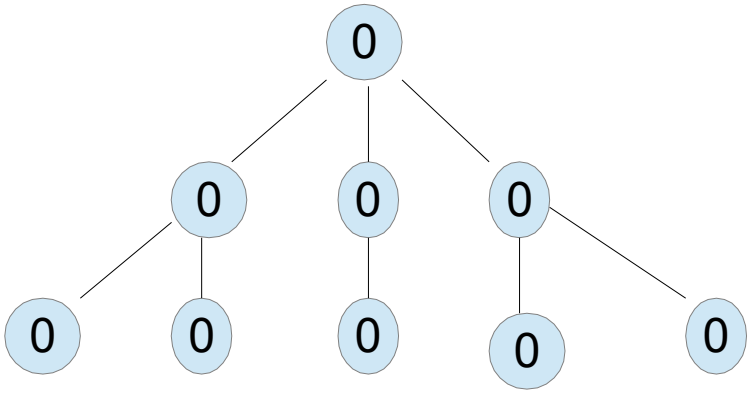
Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```



Depth first in action

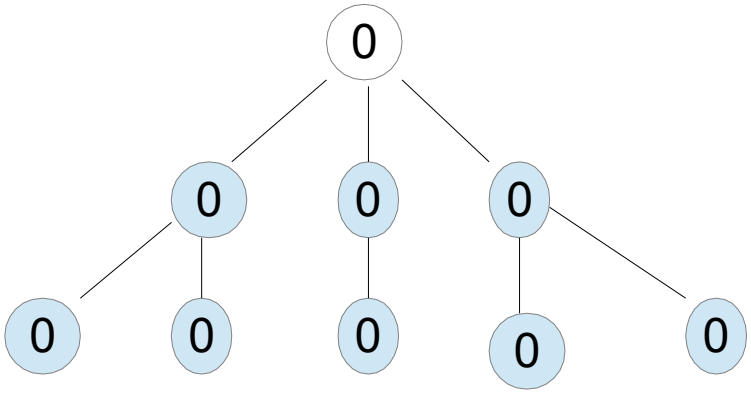
```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```



count: 0

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

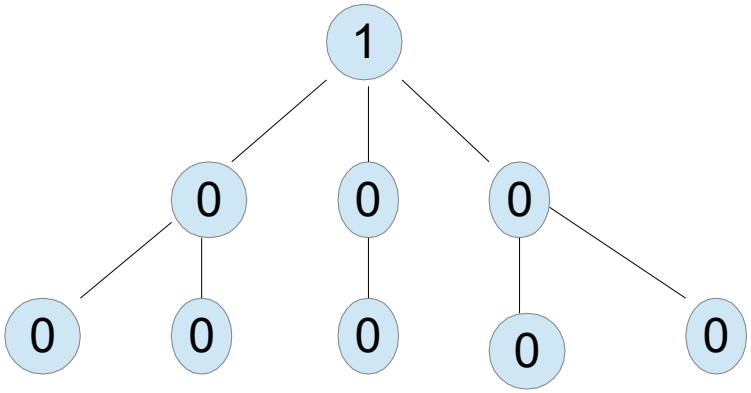


count: 0

Stack
dfs(root)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

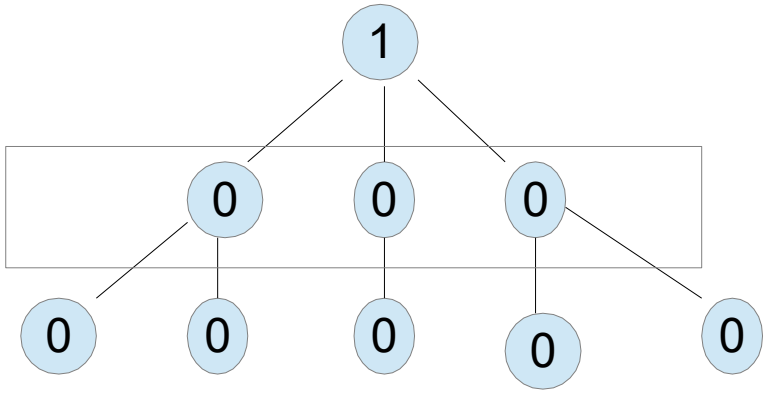


count: 1

Stack
dfs(root)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

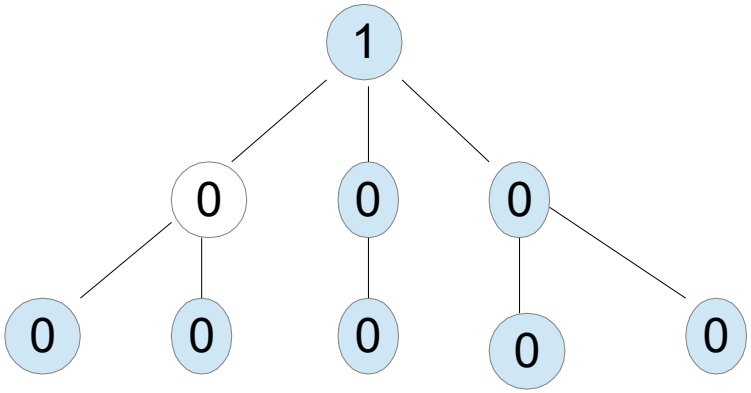


count: 1

Stack
dfs(root)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

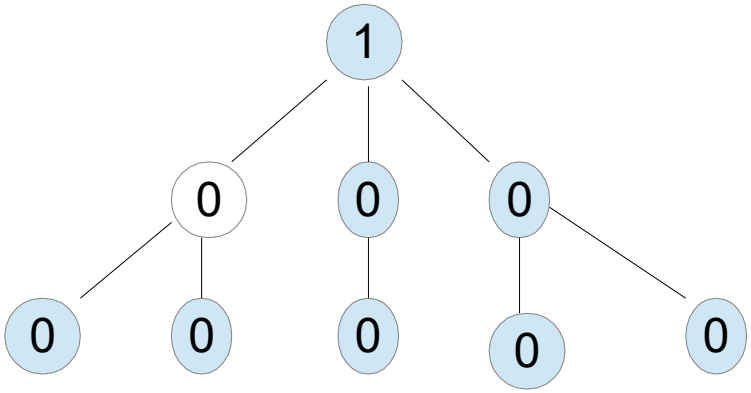


count: 1

Stack
dfs(root)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

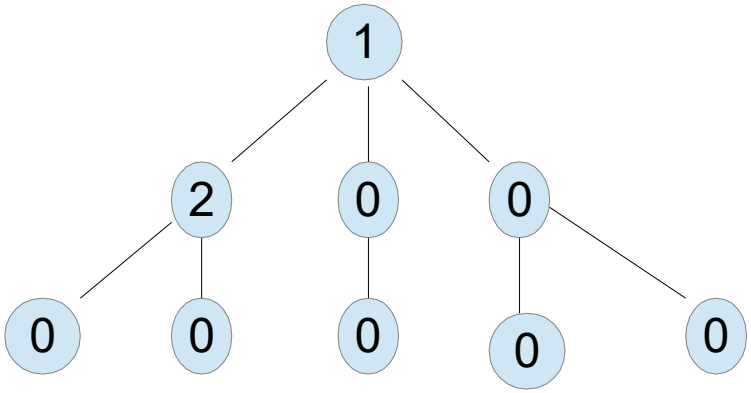


count: 1

Stack
dfs(root)
dfs(root.left)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

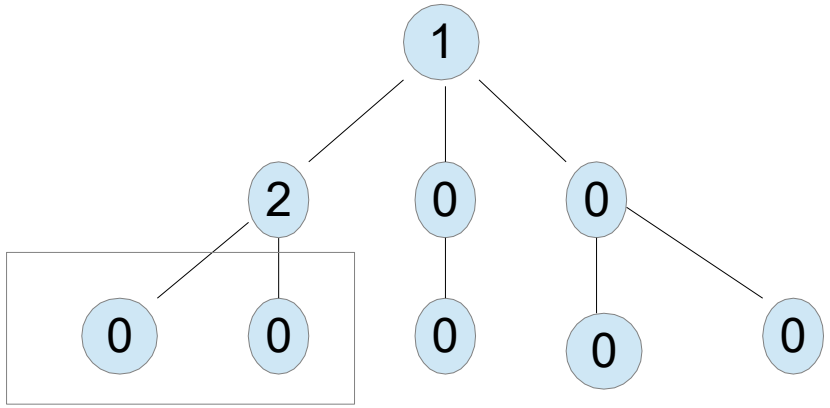


count: 2

Stack
dfs(root)
dfs(root.left)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

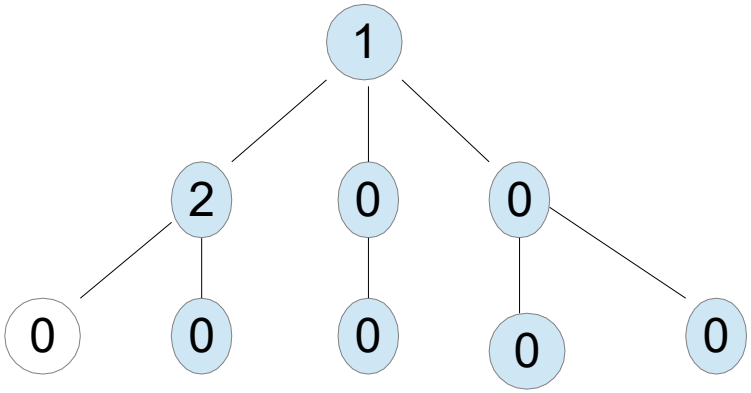


count: 2

Stack
dfs(root)
dfs(root.left)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

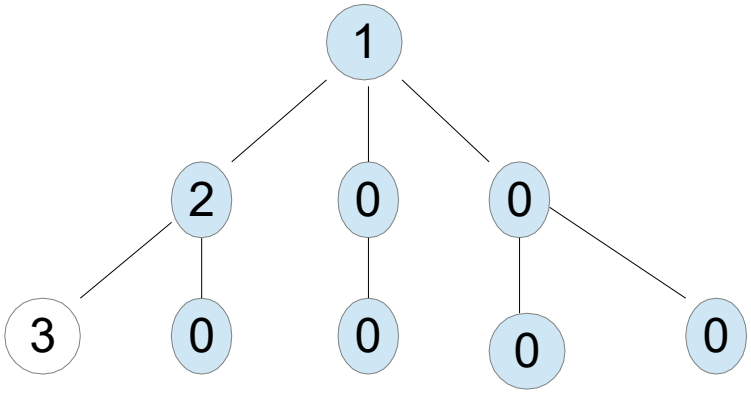


count: 2

Stack
dfs(root)
dfs(root.left)
dfs(root.left.left)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

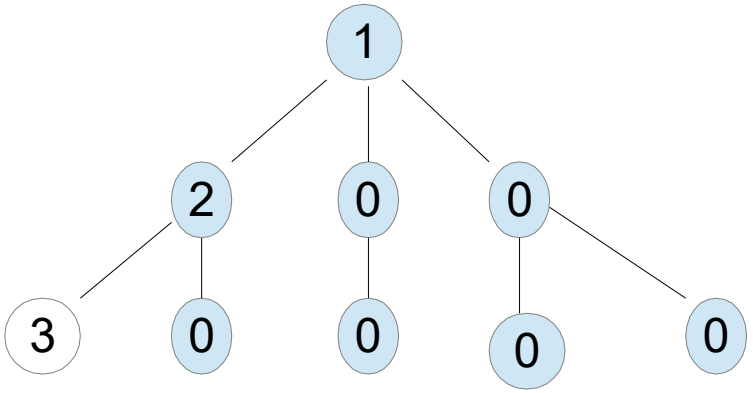


count: 3

Stack
dfs(root)
dfs(root.left)
dfs(root.left.left)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```



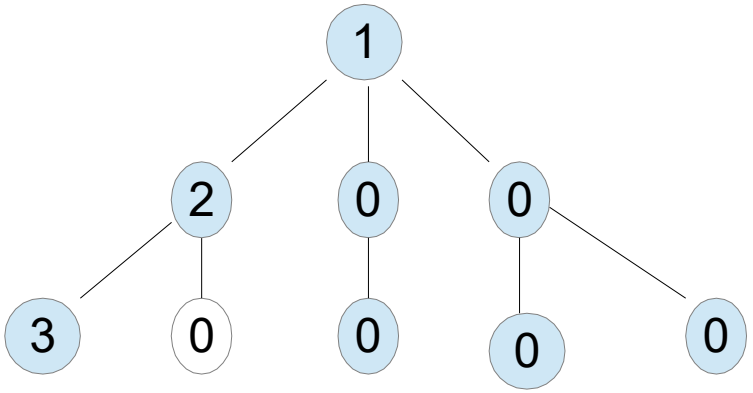
No children, so we finished dfs(root.left.left)

count: 3

Stack
dfs(root)
dfs(root.left)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

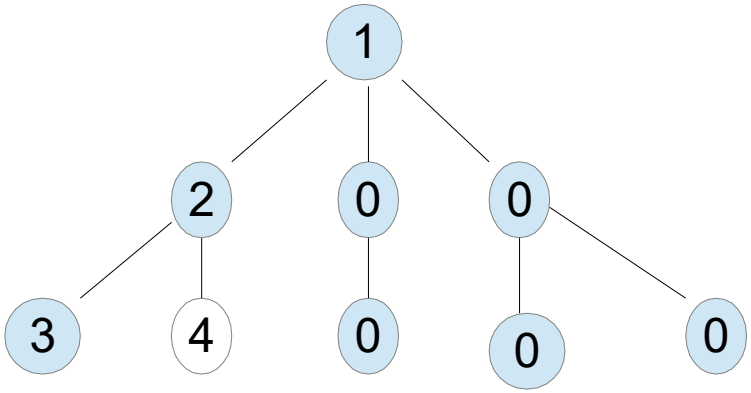


count: 4

Stack
dfs(root)
dfs(root.left)
dfs(root.left.right)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

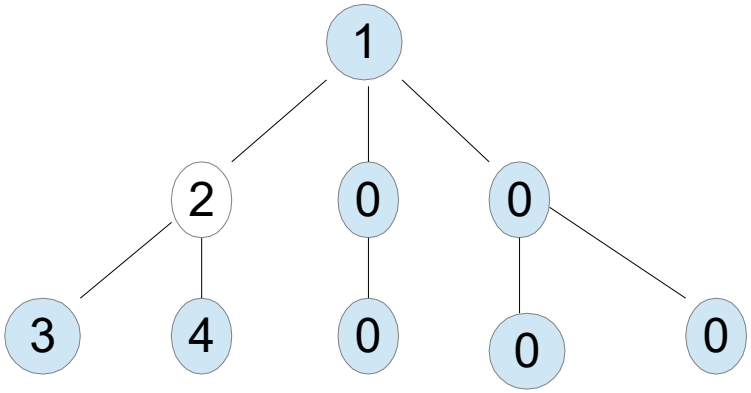


count: 4

Stack
dfs(root)
dfs(root.left)
dfs(root.left.right)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

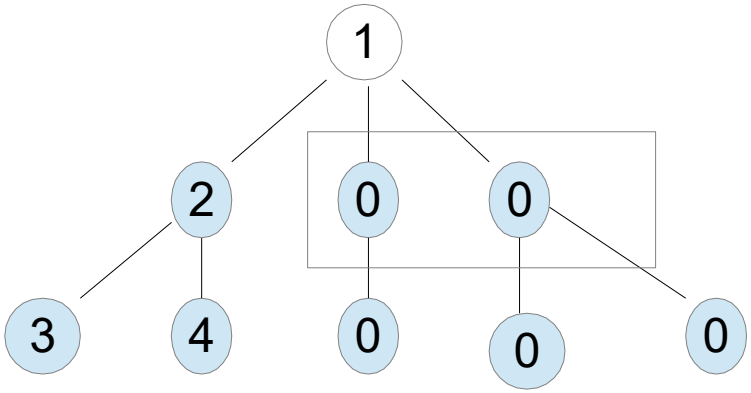


count: 4

Stack
dfs(root)
dfs(root.left)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

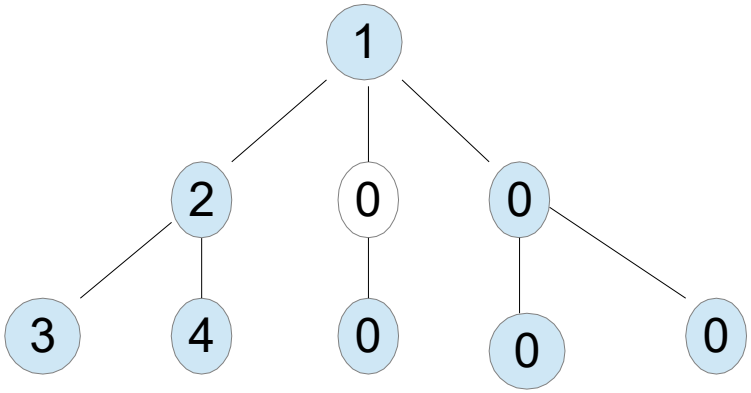


count: 4

Stack
dfs(root)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```

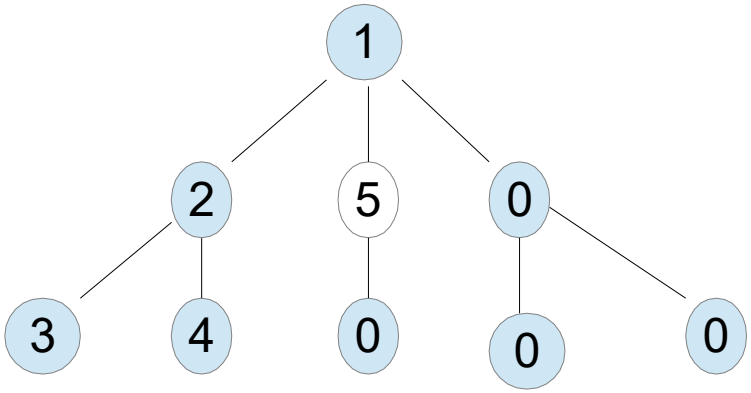


count: 4

Stack
dfs(root)
dfs(root.middle)

Depth first in action

```
DepthFirst(G, V, E):
    mark all vertices with 0
    count = 0
    for each vertex v in V:
        if v is marked with 0:
            dfs(v)

dfs(v):
    ++count;
    mark v with count
    for each vertex w adjacent to v:
        if w is marked with a 0:
            dfs(w)
```



count: 5

Stack
dfs(root)
dfs(root.middle)

Etc.

It was easy to illustrate with a tree-like structure.

But depth-first and breadth-first work with any graph.

I named the vertices root, root.left etc for convenience, not because these are tree algorithms. They are graph algorithms.

Depth first search gives two orders

- Order in which nodes were visited
  - Represented by the variable called ***count***
- Order in which vertices were **popped** off the stack
  - In our example that was:
    - Root.left.left then root.left.right then root.left etc.

Basic Breadth First

BreadthFirst is identical to DepthFirst except it calls bfs instead of dfs.

```
bfs(v):
    ++count
    mark v with count
    create a new queue and insert v
    while the queue is not empty:
        for each vertex w adjacent to front vertex
    if w is marked with 0:
                ++count
                mark w with count
                add w to the queue
        remove the front vertex from the queue
```

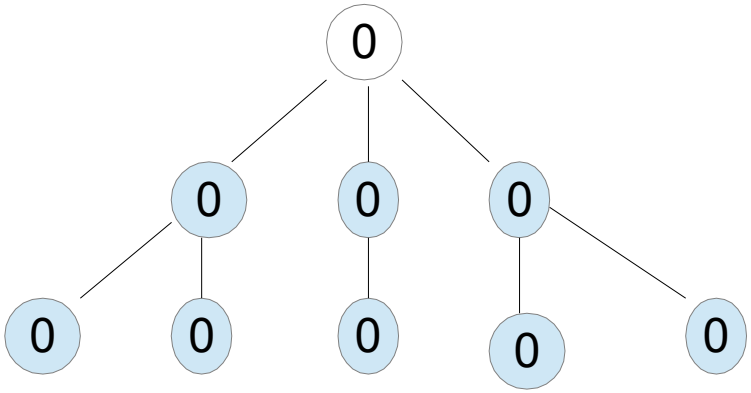
Breadth first in action

BreadthFirst

…

bfs(v):

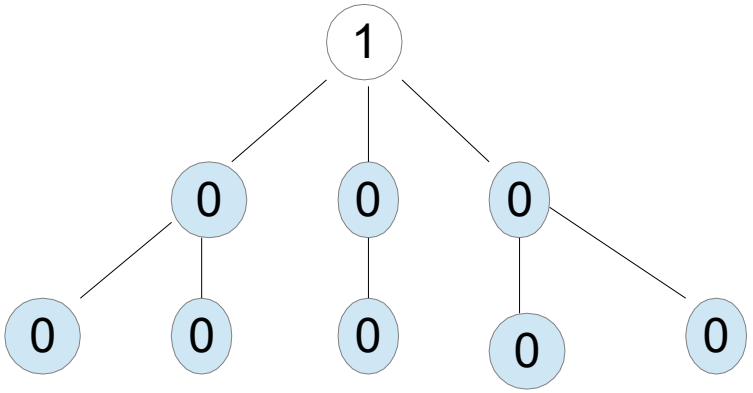
```
++count
mark v with count
create a new queue and insert v
while the queue is not empty:
    for each vertex w adjacent
  to front vertex
        if w is marked with 0:
            ++count
            mark w with count
            add w to the queue
    remove the front vertex from the queue
```



count: 0

Breadth first in action

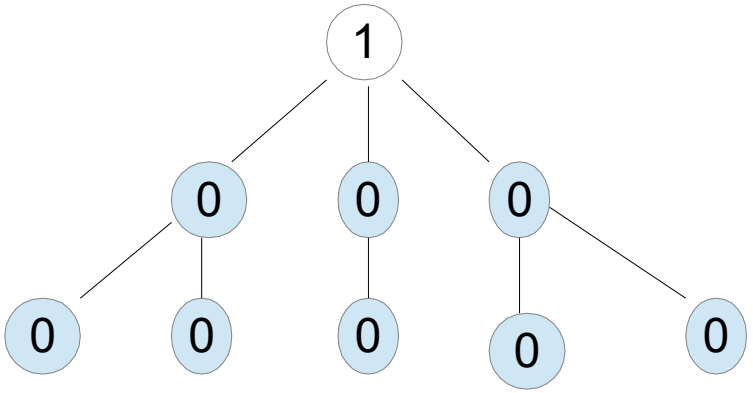
```
bfs(v):
    ++count
    mark v with count
    create a new queue and insert v
    while the queue is not empty:
        for each vertex w adjacent
      to front vertex
            if w is marked with 0:
                ++count
                mark w with count
                add w to the queue
        remove the front vertex from the queue
```



count: 1

Breadth first in action

```
bfs(v):
    ++count
    mark v with count
    create a new queue and insert v
    while the queue is not empty:
        for each vertex w adjacent
      to front vertex
            if w is marked with 0:
                ++count
                mark w with count
                add w to the queue
        remove the front vertex from the queue
```

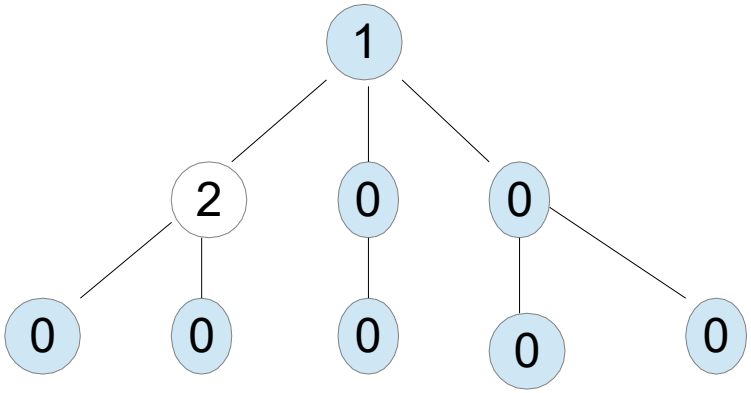


count: 1

Queue: root

Breadth first in action

```
bfs(v):
    ++count
    mark v with count
    create a new queue and insert v
    while the queue is not empty:
        for each vertex w adjacent
        to front vertex
            if w is marked with 0:
                ++count
                mark w with count
                add w to the queue
        remove the front vertex from the queue
```

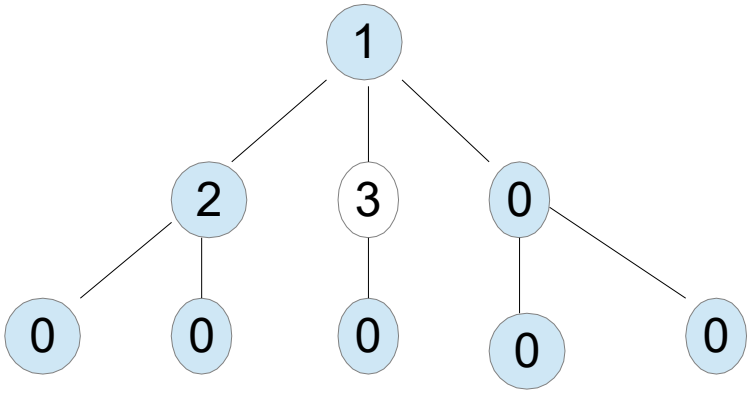


count: 2

Queue: root **root.left**

Breadth first in action

```
bfs(v):
    ++count
    mark v with count
    create a new queue and insert v
    while the queue is not empty:
        for each vertex w adjacent
        to front vertex
            if w is marked with 0:
                ++count
                mark w with count
                add w to the queue
        remove the front vertex from the queue
```

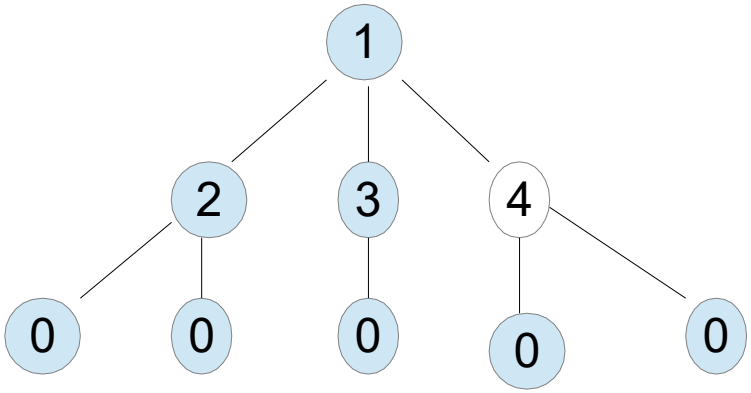


count: 3

Queue: root root.left **root.middle**

Breadth first in action

```
bfs(v):
    ++count
    mark v with count
    create a new queue and insert v
    while the queue is not empty:
        for each vertex w adjacent
      to front vertex
            if w is marked with 0:
                ++count
                mark w with count
                add w to the queue
        remove the front vertex from the queue
```

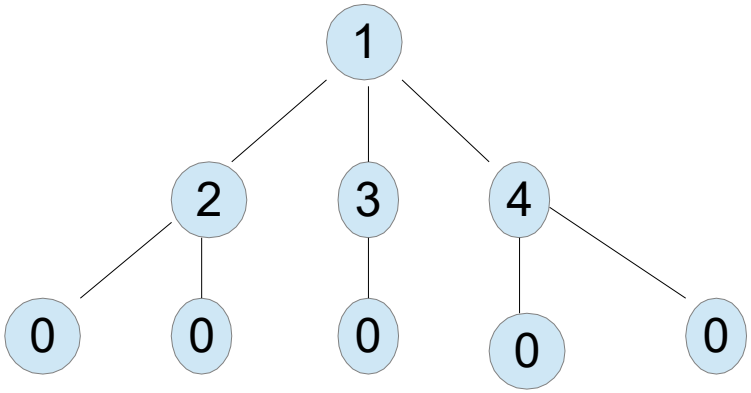


count: 4

Queue: root root.left root.middle **root.right**

Breadth first in action

```
bfs(v):
    ++count
    mark v with count
    create a new queue and insert v
    while the queue is not empty:
        for each vertex w adjacent
      to front vertex
            if w is marked with 0:
                ++count
                mark w with count
                add w to the queue
       remove the front vertex from the queue
```

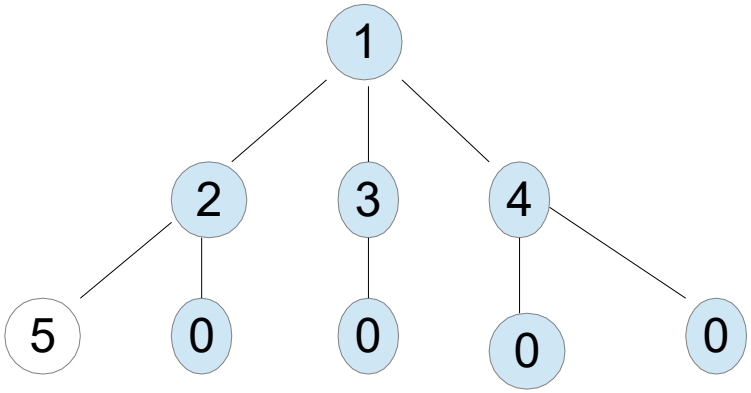


count: 4

Queue: root.left root.middle root.right
Note: root removed from Queue

Breadth first in action

```
bfs(v):
    ++count
    mark v with count
    create a new queue and insert v
    while the queue is not empty:
        for each vertex w adjacent
      to front vertex
            if w is marked with 0:
                ++count
                mark w with count
                add w to the queue
        remove the front vertex from the queue
```



count: 5

Queue: root.left root.middle root.right **root.left.left**

Etc.