応用アルゴリズム演習

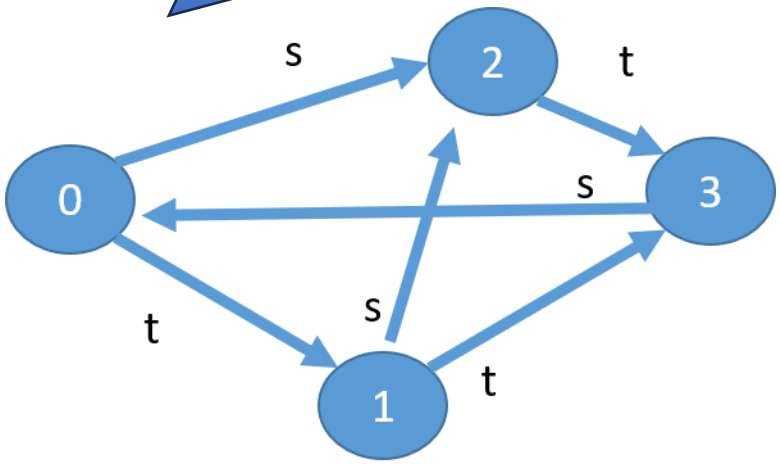
Depth First Search　/　深さ優先探索

補足

2023/11/2

課題１・問題１の回答

s の枝が先に探索される

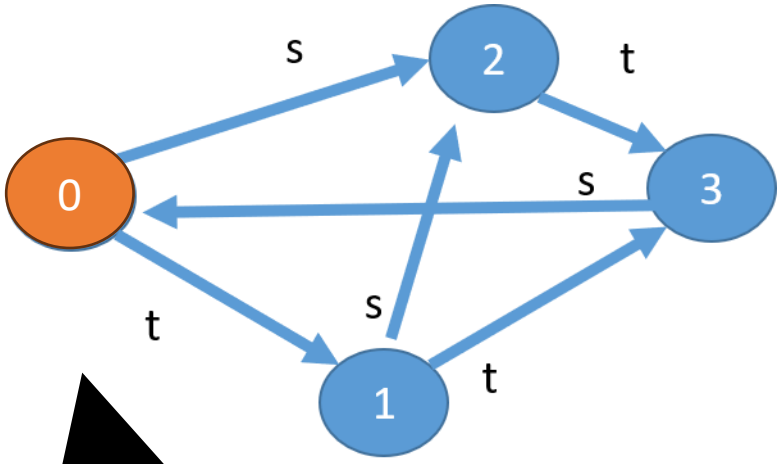



```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

(node 0, visited 0 times) // stack:0
(node 2, visited 0 times) // stack:0:2
(node 3, visited 0 times) // **stack:0:2:3**
(node 0, visited 1 times) // **stack:0:2:3:0**
(node 1, visited 0 times) // **stack:0:1**
(node 2, visited 1 times) // **stack:0:1:2**
(node 3, visited 1 times) // **stack:0:1:3**

課題１・問題１の回答

s の枝が先に探索される



(node 0, visited 0 times)

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

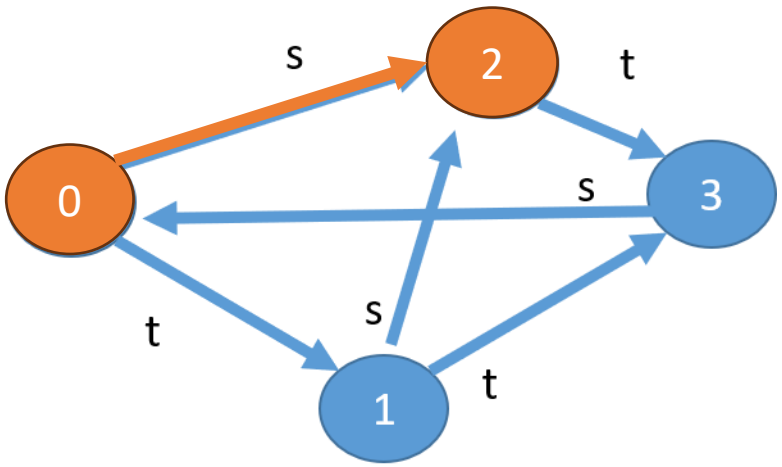
Call Stack:

0

課題１・問題１の回答

s の枝が先に探索される

(node 2, visited 0 times)



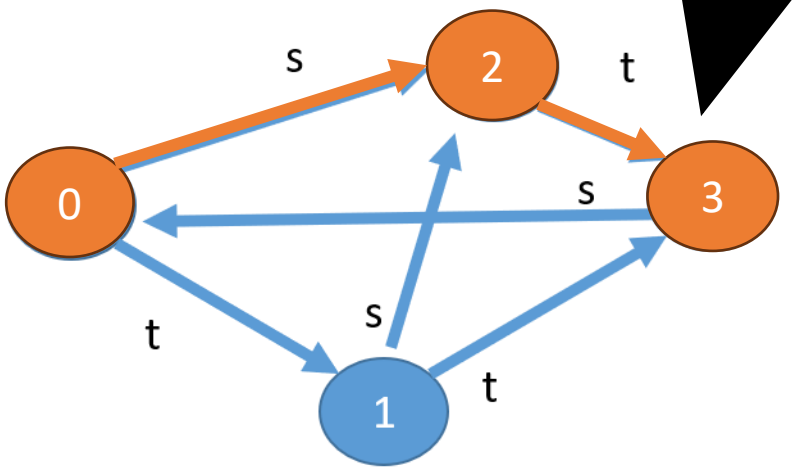

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Call Stack:

課題１・問題１の回答

s の枝が先に探索される

(node 3, visited 0 times)



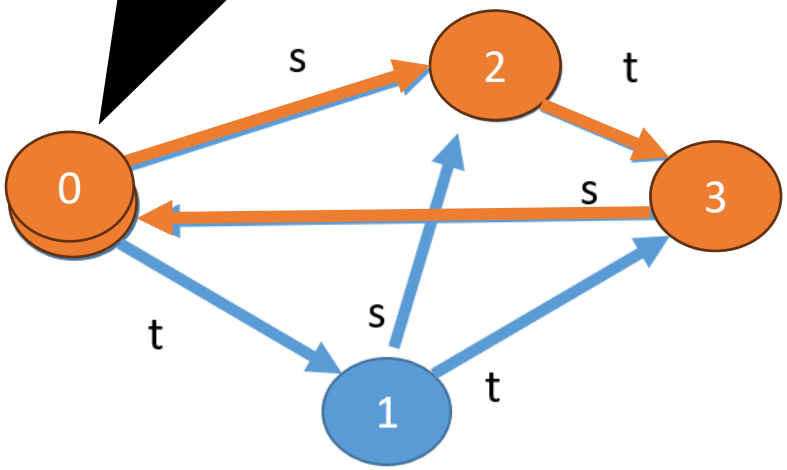

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Call Stack:

課題１・問題１の回答

s の枝が先に探索される

(node 3, visited 1 times)




```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

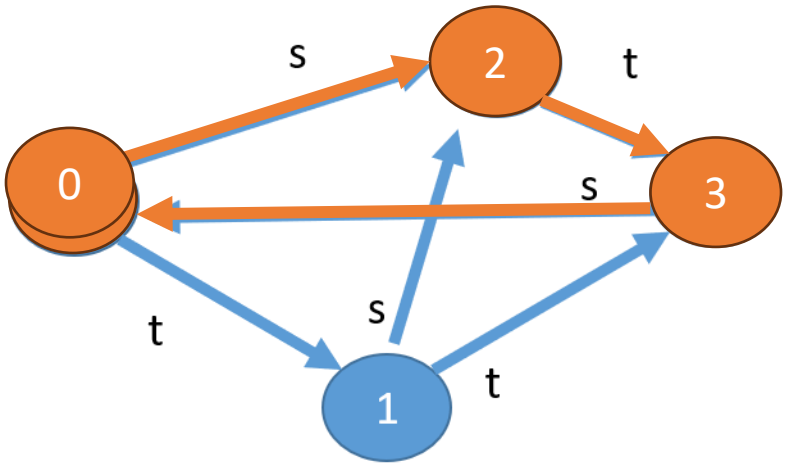
Call Stack:

課題１・問題１の回答

Node 0　訪問するのが初めてじゃない！それ以上探索しない

s の枝が先に探索される

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```



Call Stack:

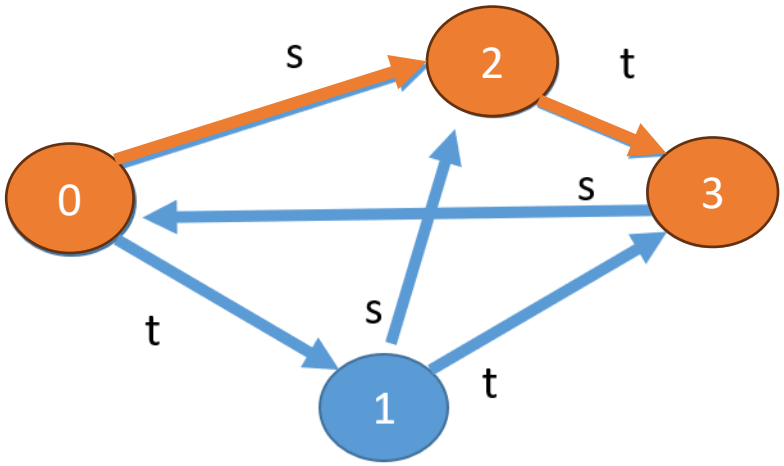
0	2	3	0

課題１・問題１の回答

s の枝が先に探索される

Node 0　訪問するのが初めてじゃない！
それ以上探索しない

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```



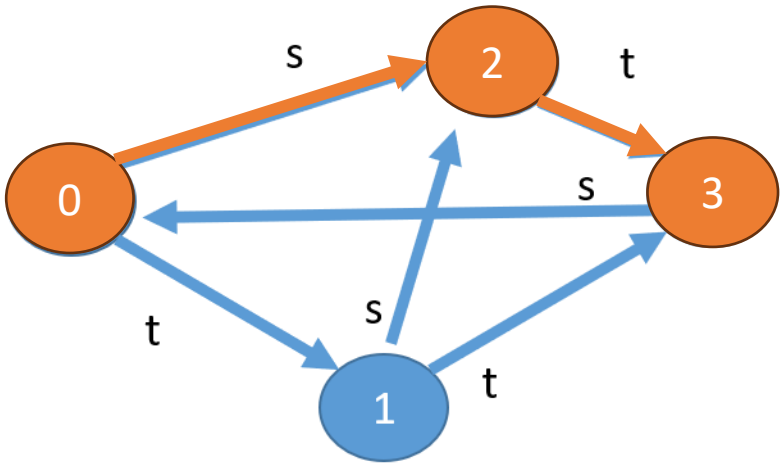
Call Stack:

課題１・問題１の回答

s の枝が先に探索される

Node 0　訪問するのが初めてじゃない！
それ以上探索しない

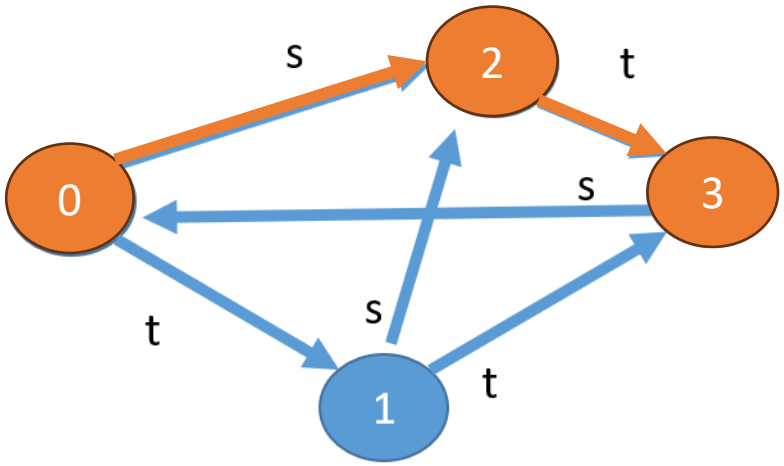
```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```



Call Stack:

課題１・問題１の回答

s の枝が先に探索される



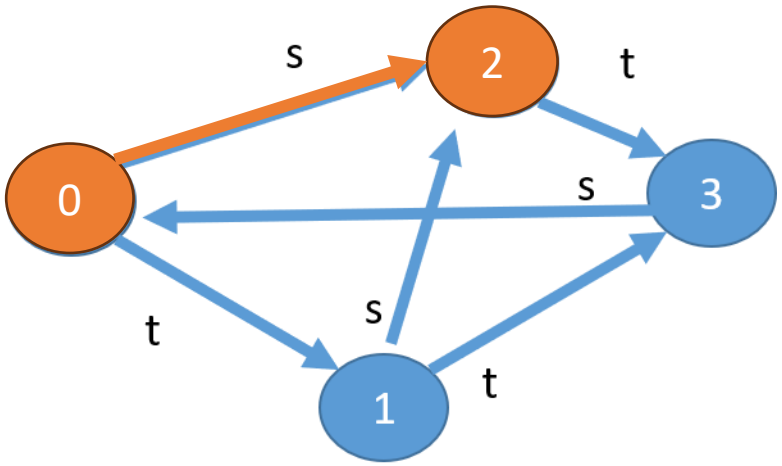

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 3 に隣接するノードが他にありません

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



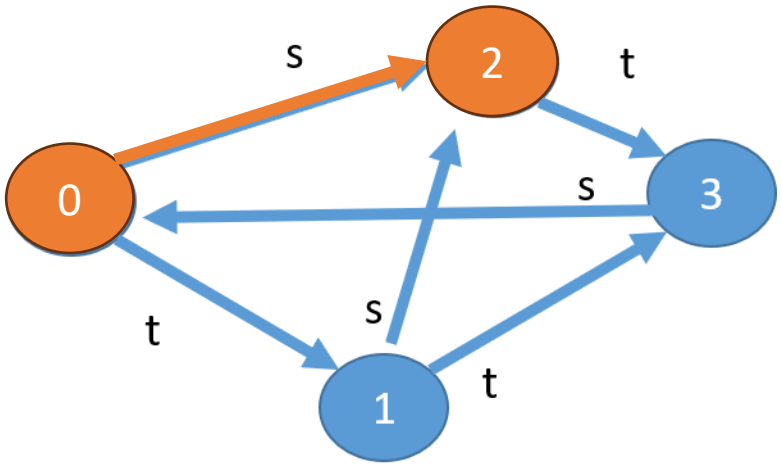

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 3 に隣接するノードが他にありません

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



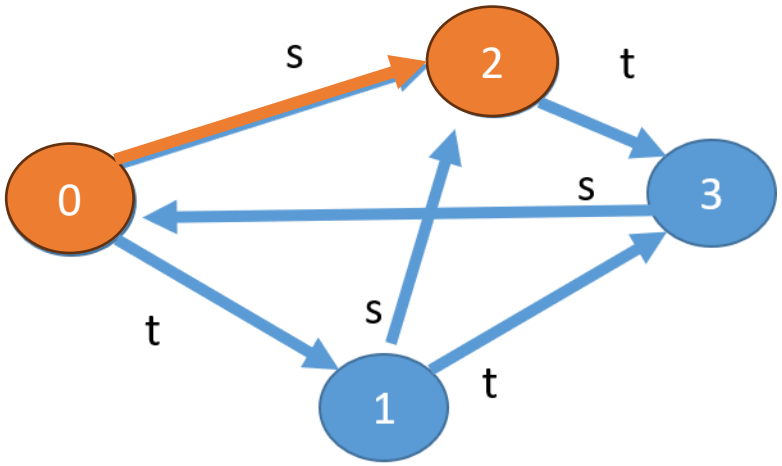

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 3 に隣接するノードが他にありません

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



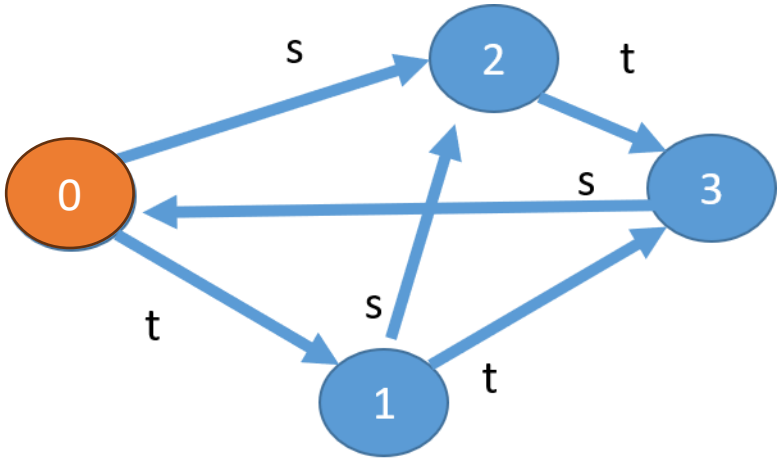

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 2 に隣接するノードが他にありません

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



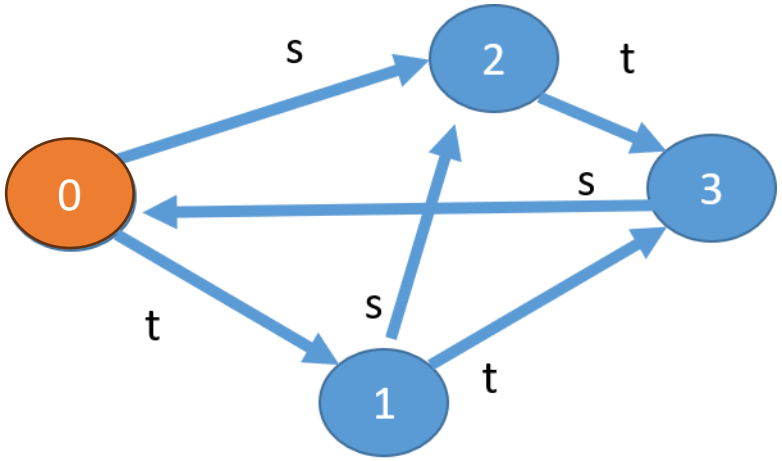

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 2 に隣接するノードが他にありません

Call Stack:

課題１・問題１の回答

s の枝が先に探索される




```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

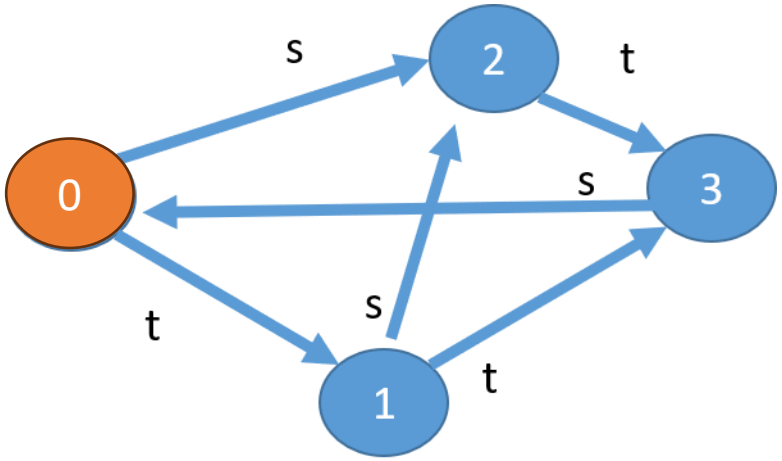
Node 2 に隣接するノードが他にありません

Call Stack:

0

課題１・問題１の回答

s の枝が先に探索される



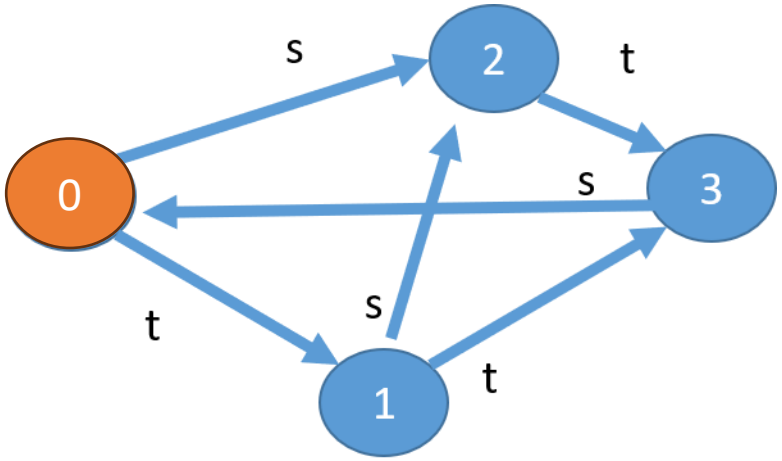

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Call Stack:

0

課題１・問題１の回答

s の枝が先に探索される




```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 0 の枝 t を渡って，
ノード１へ行きます！

Call Stack:

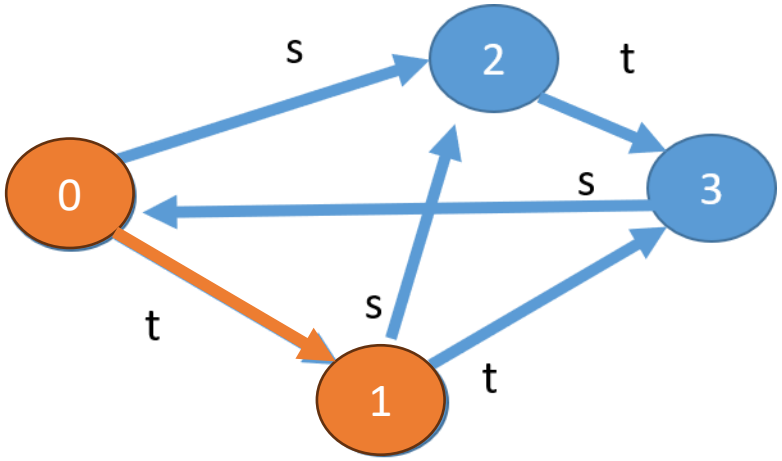
0

課題１・問題１の回答

s の枝が先に探索される

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 0 の枝 t を渡って，
ノード１へ行きます！



(node 1, visited 0 times)

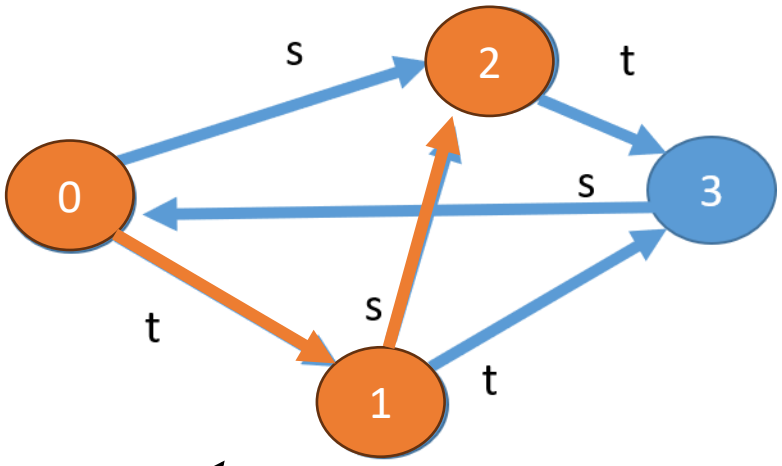
Call Stack:

課題１・問題１の回答

s の枝が先に探索される

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 1 の枝 s を渡って，
ノード2へ行きます！

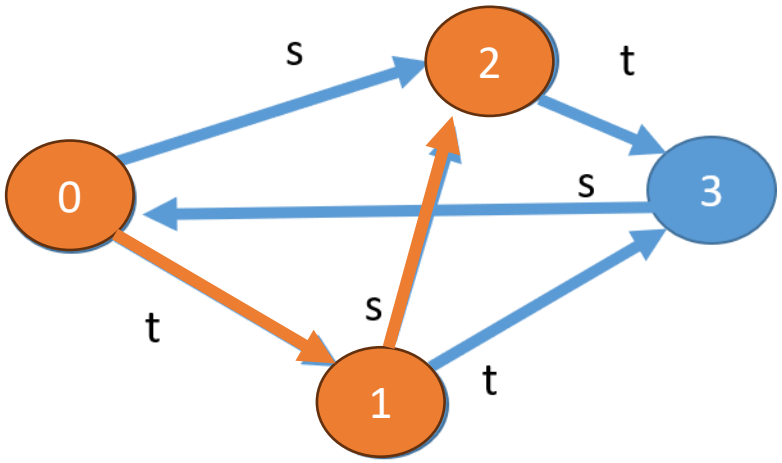


(node 2, visited 1 times)

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



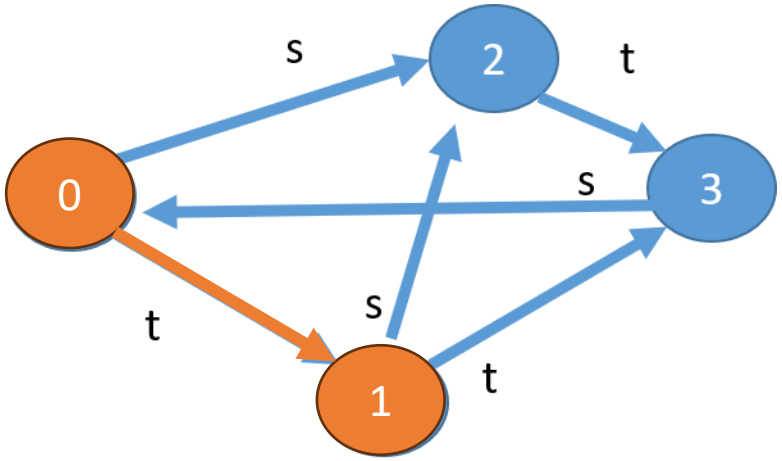

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 2　訪問するのが初めてじゃない！
それ以上探索しない

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



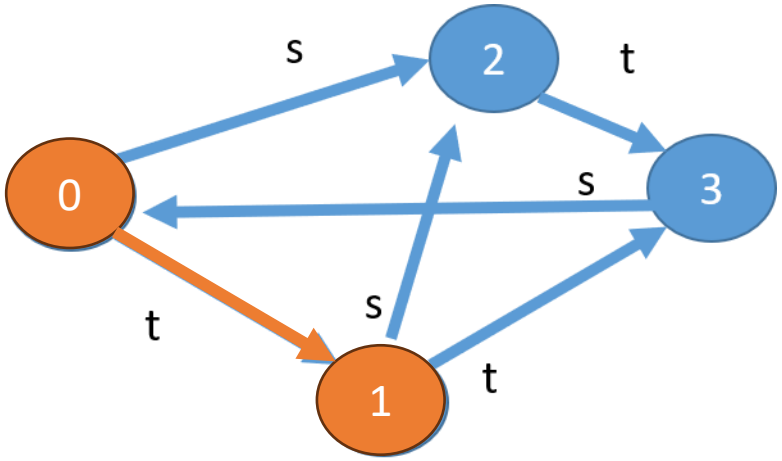

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 2　訪問するのが初めてじゃない！
それ以上探索しない

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



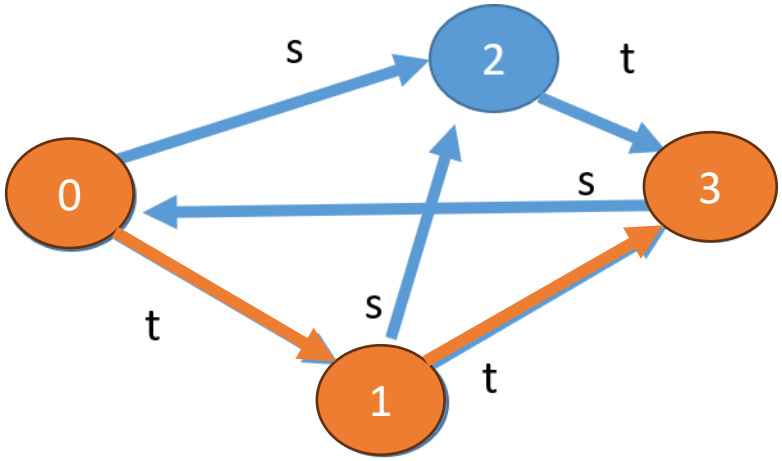

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 2　訪問するのが初めてじゃない！
それ以上探索しない

Call Stack:

課題１・問題１の回答

s の枝が先に探索される




```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

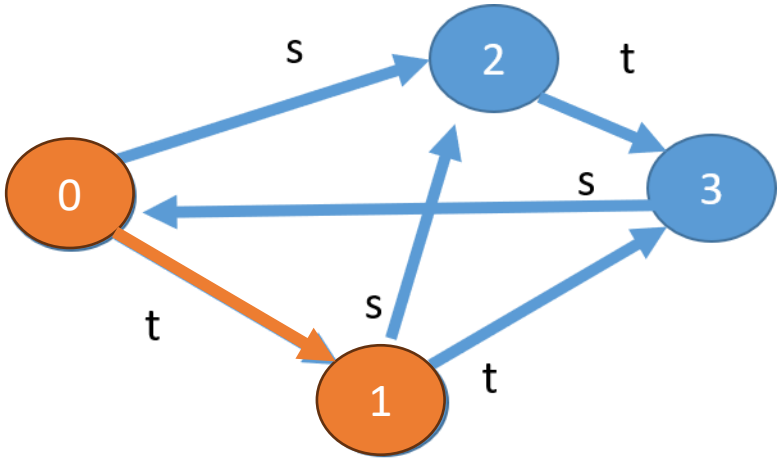
Node 1 の枝 t を渡って，
ノード3へ行きます！

(node 3, visited 1 times)

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



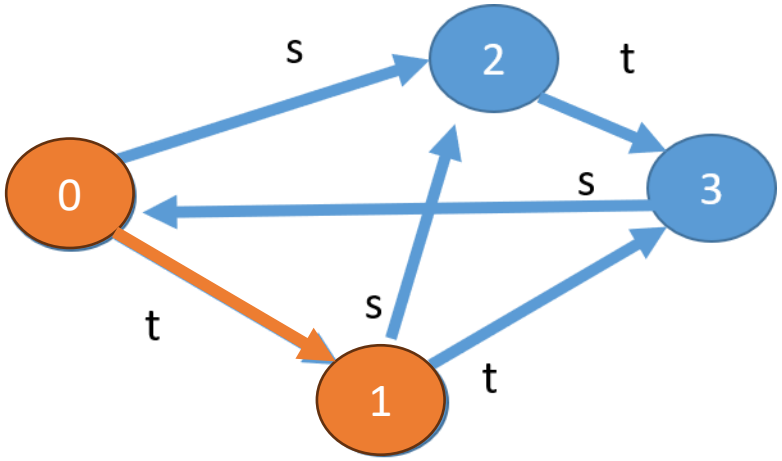

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 3 訪問するのが初めてじゃないから，それ以上探索しない

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



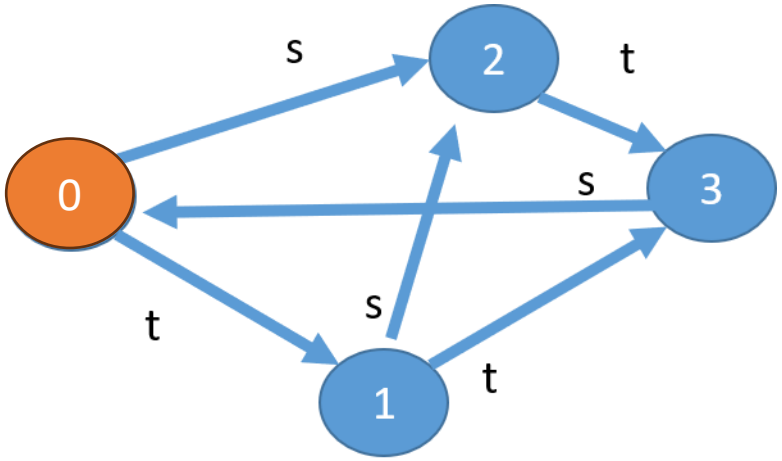

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 3 訪問するのが初めてじゃないから，それ以上探索しない

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



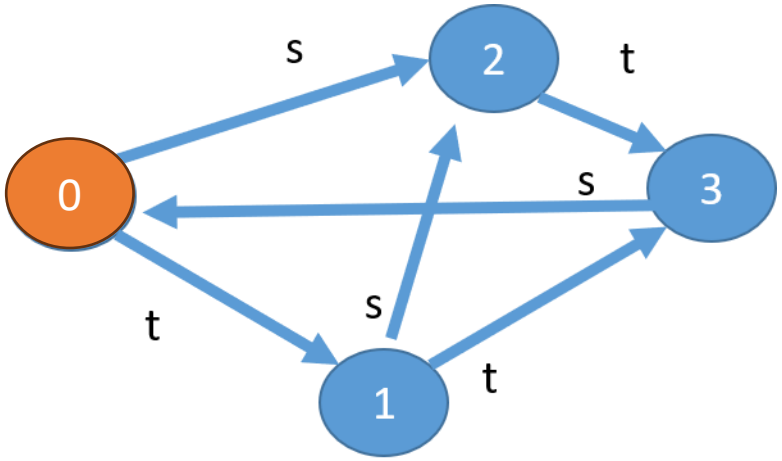

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 1 に枝が他にありません

Call Stack:

課題１・問題１の回答

s の枝が先に探索される



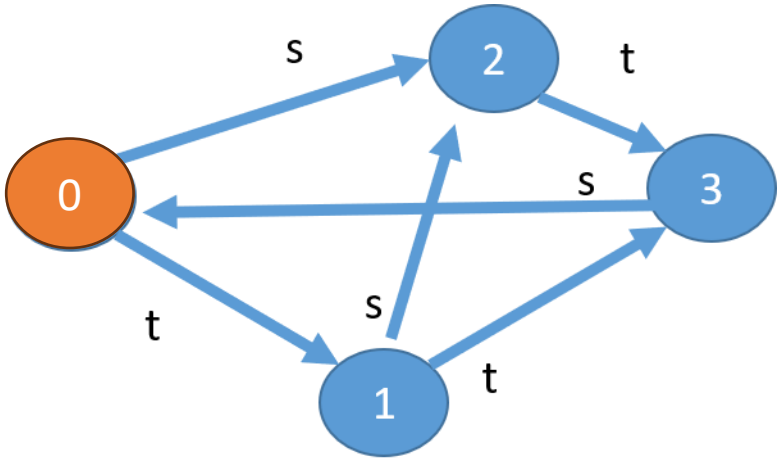

```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Call Stack:

0

課題１・問題１の回答

s の枝が先に探索される




```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

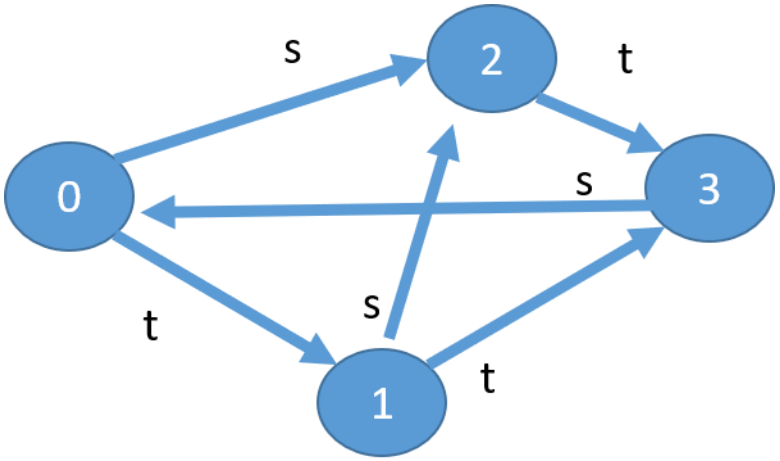
Node 0 に枝が他にありません

Call Stack:

0

課題１・問題１の回答

s の枝が先に探索される




```
void dfs(node_t * node) {
    printNode(node);
    node->visited++;
    if (node->visited == 1) {
        int n;
        for (n = 0; n < node->neighborCount; n++) {
            node_t * next = node->neighbors[n];
            dfs(next);
        }
    }
}
```

Node 0 に枝が他にありません

Call Stack:

ø

===探索終了===

課題１　再提出

- **test3() で実装されたグラフを利用**
- **問題１と同じ内容**

```
(node 0, visited 0 times) stack: 0
(node 3, visited 0 times) stack: ...
(node 1, visited 0 times)
(node 2, visited 0 times)
(node 3, visited 1 times)
(node 3, visited 2 times)
```

s の枝が先に探索される