Depth First Search (DFS)

- Idea: Explore a graph by exploring as far as possible along each branch before backtracking
- How it works:

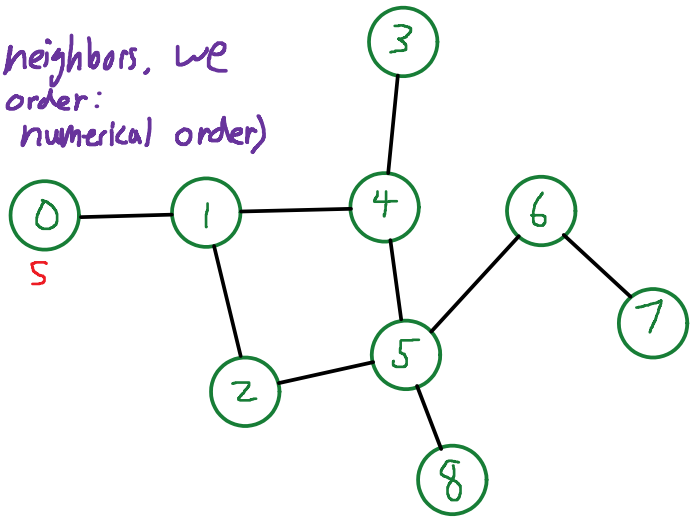
```
DFS-iterative(root):
    let Stk be stack
    let visited be set
    Stk.push(root)
    while (Stk is not empty):
        v = Stk.pop()
        for all neighbors w of v in graph:
            if w not in visited:
                Stk.push(w)
                visited.add(w)
```

- Example: Assume if we have 2+ neighbors, we want to visit each by numerical order: add neighbors to stack in reverse numerical order)

Stk:
[ ]

visited:
{ }

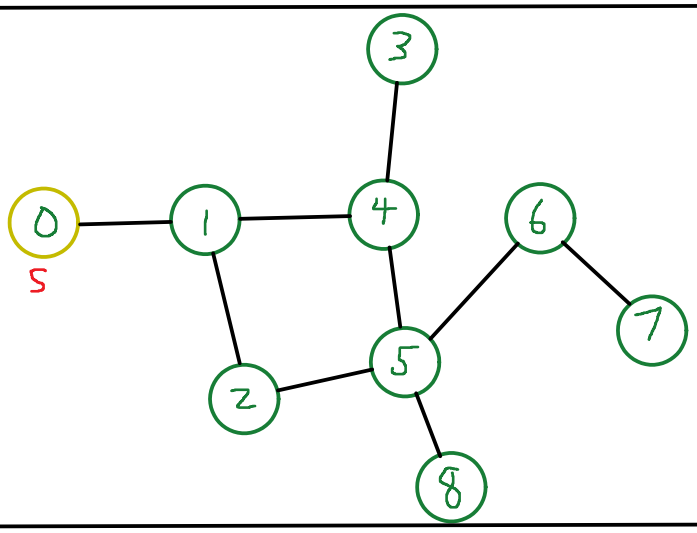
Preorder:



Stk:
[0]

visited:
{0}

Preorder:
0

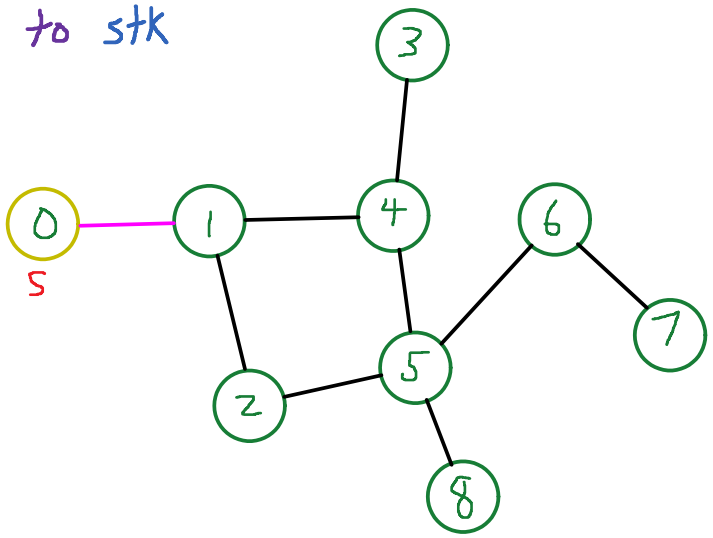


Pop off 0, add neighbors to stk

Stk:
[1]

Visited:
{0}

PreOrder:
0

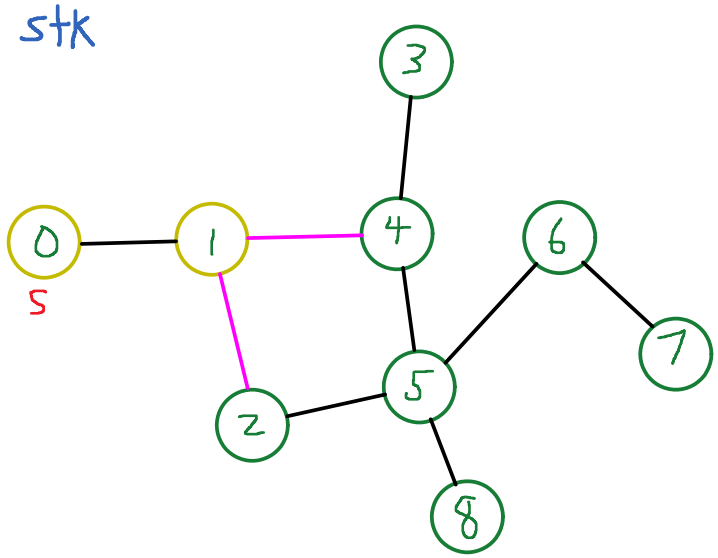


Pop off 1, add neighbors to stk

Stk: ← top of stack
[4 2]

Visited:
{0 1}

PreOrder:
0 1

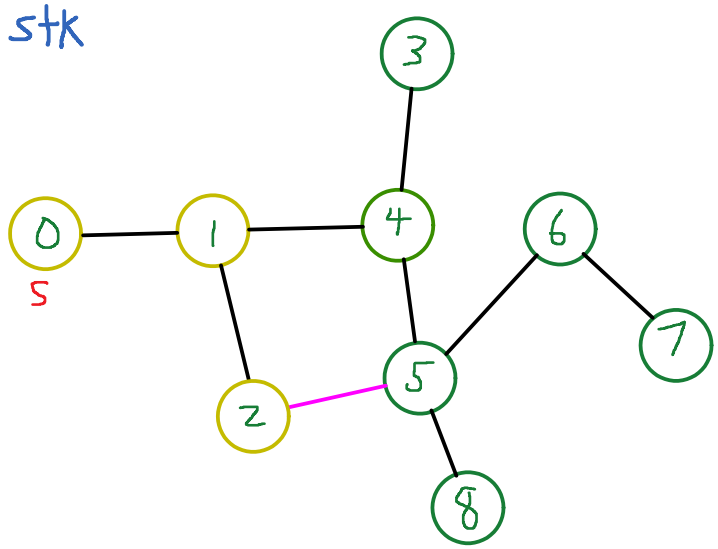


Pop off 2, add neighbors to stk

Stk
[4 5]

Visited:
{0 1 2}

PreOrder:
0 1 2



Pop off 5, add neighbors to stk

Stk:

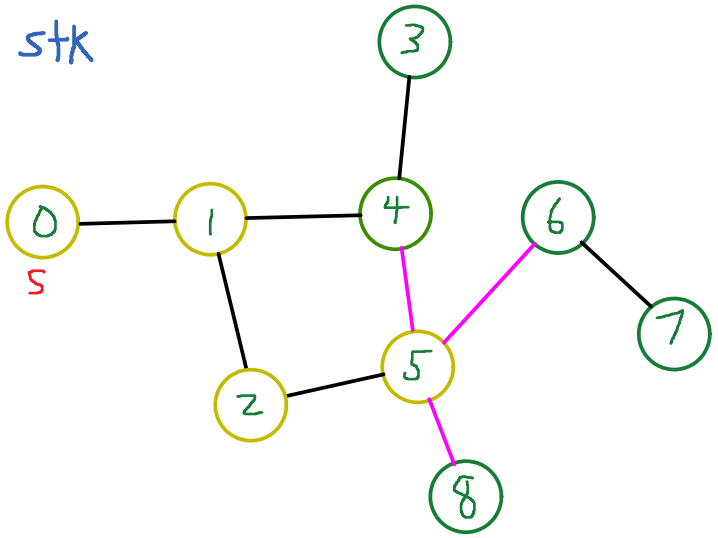
[4 8 6 4]

Visited:

{0 1 2 5}

PreOrder:

0 1 2 5



Pop off 4, add neighbors to stk

Stk:

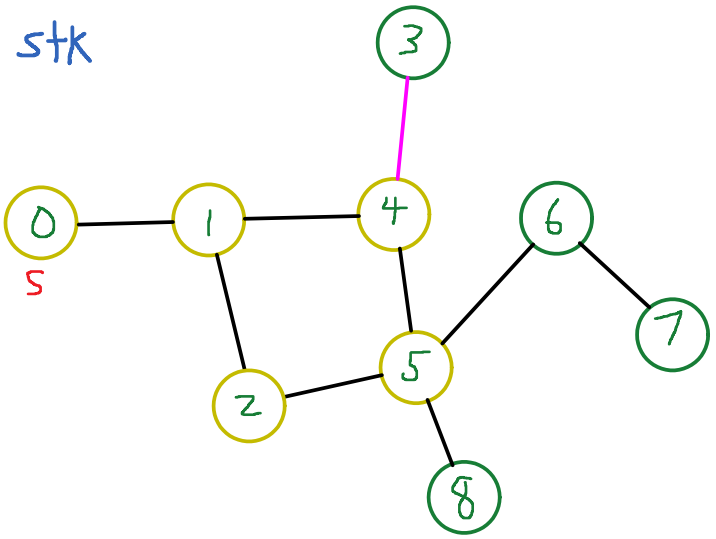
[4 8 6 3]

Visited:

{0 1 2 5 4}

PreOrder:

0 1 2 5 4



Pop off 3, add neighbors to stk (none)

Stk:

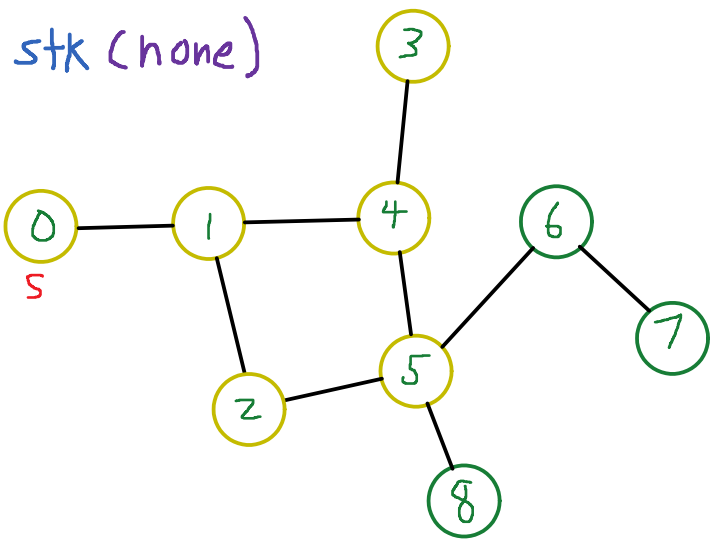
[4 8 6]

Visited:

{0 1 2 5 4 3}

PreOrder:

0 1 2 5 4 3



Pop off 6, add neighbors to stk

Stk:

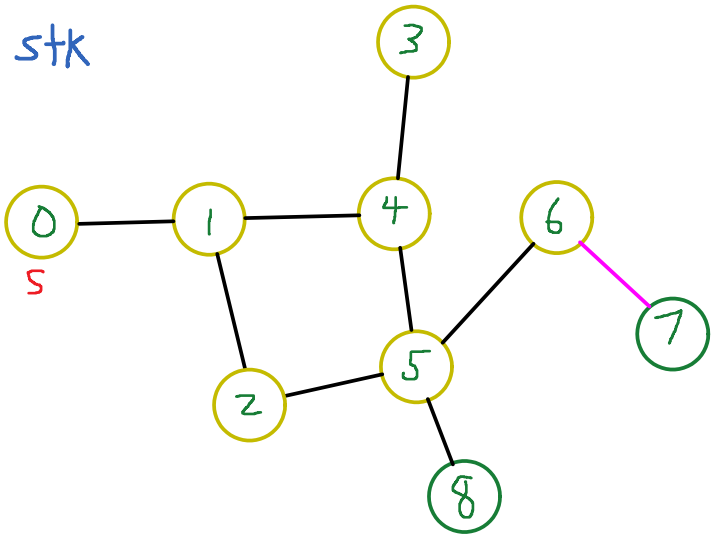
[4 8 7]

Visited:

{0 1 2 5 4 3 6}

PreOrder:

0 1 2 5 4 3 6



Pop off 6, add neighbors to stk

Stk:

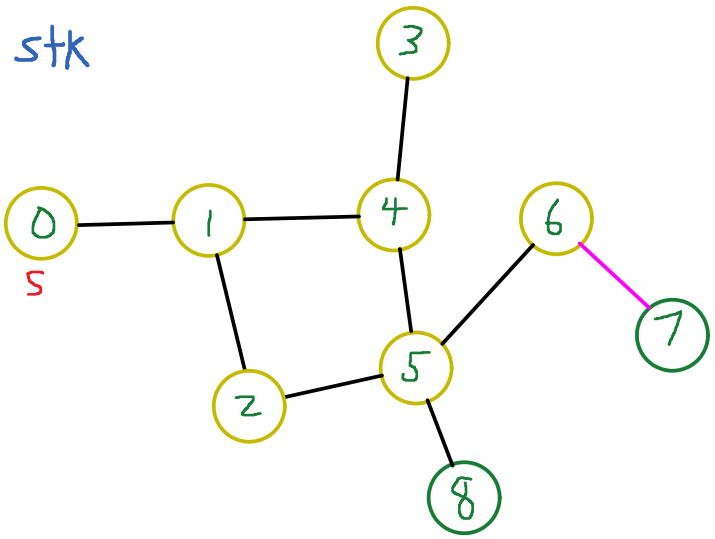
[4 8 7]

Visited:

{0 1 2 5 4 3 6}

PreOrder:

0 1 2 5 4 3 6



Pop off 7, add neighbors to stk (none)

Stk:

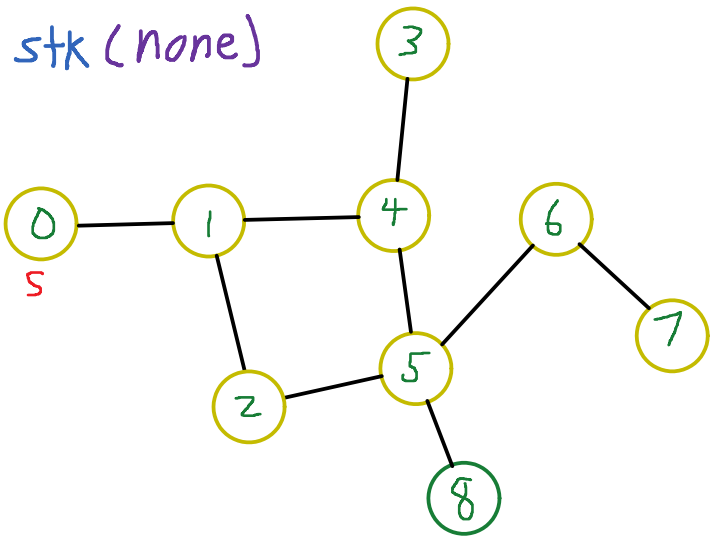
[4 8]

Visited:

{0 1 2 5 4 3 6 7}

PreOrder:

0 1 2 5 4 3 6 7



Pop off 8, add neighbors to stk (none)

Stk:

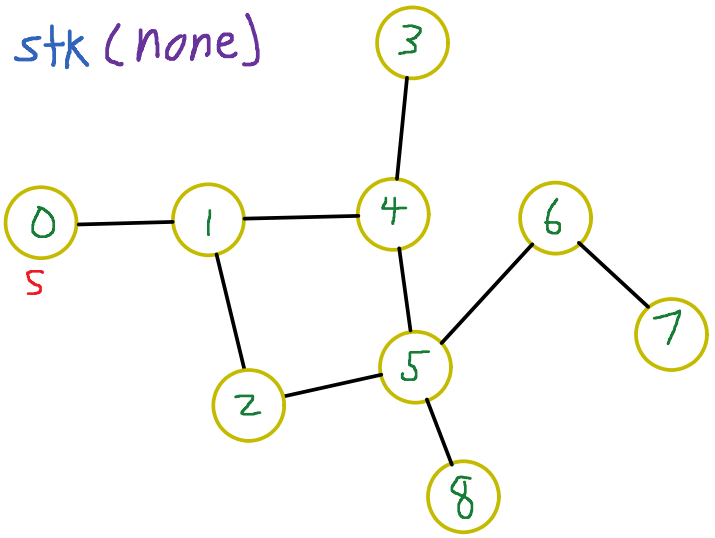
[4]

Visited:

{0 1 2 5 4 3 6 7 8}

PreOrder:

0 1 2 5 4 3 6 7 8



Pop off 4, in Visited so continue

Stk:

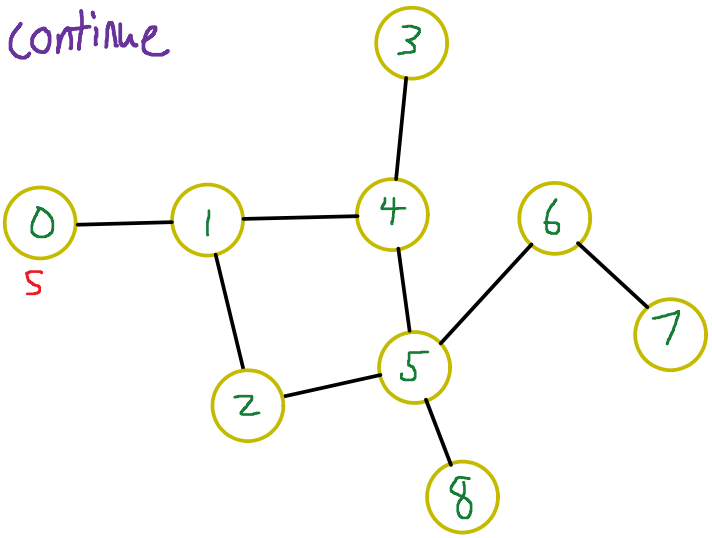
[]

Visited:

{0 1 2 5 4 3 6 7 8}

PreOrder:

0 1 2 5 4 3 6 7 8



Stk empty, so done