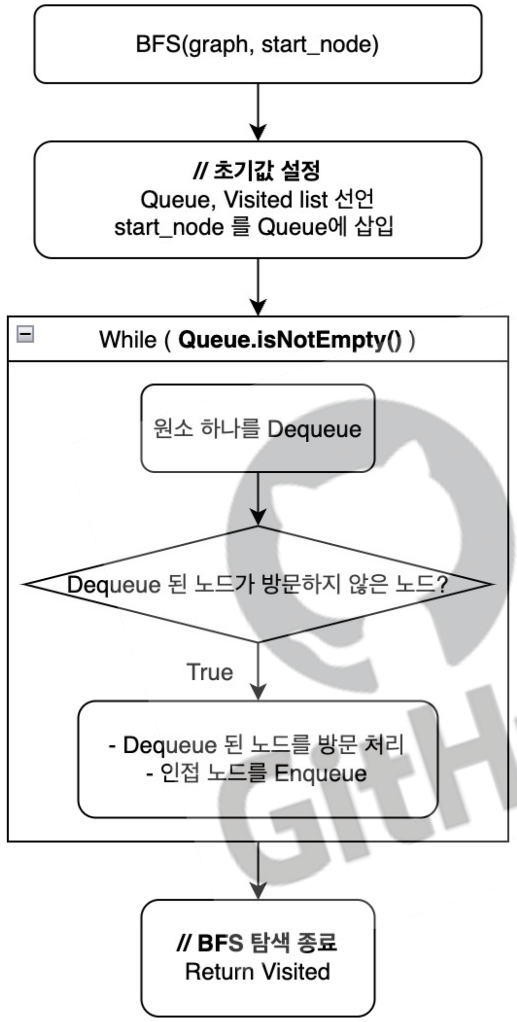
last update: 23.01.10

Graph Traversal - BFS (Breadth First Search)

Woody K
Contact: woody35545@gmail.com
Github: https://github.com/woody35545
CNU, Computer Science and Engineering

Graph Traversal – **BFS**(Breadth First Search)

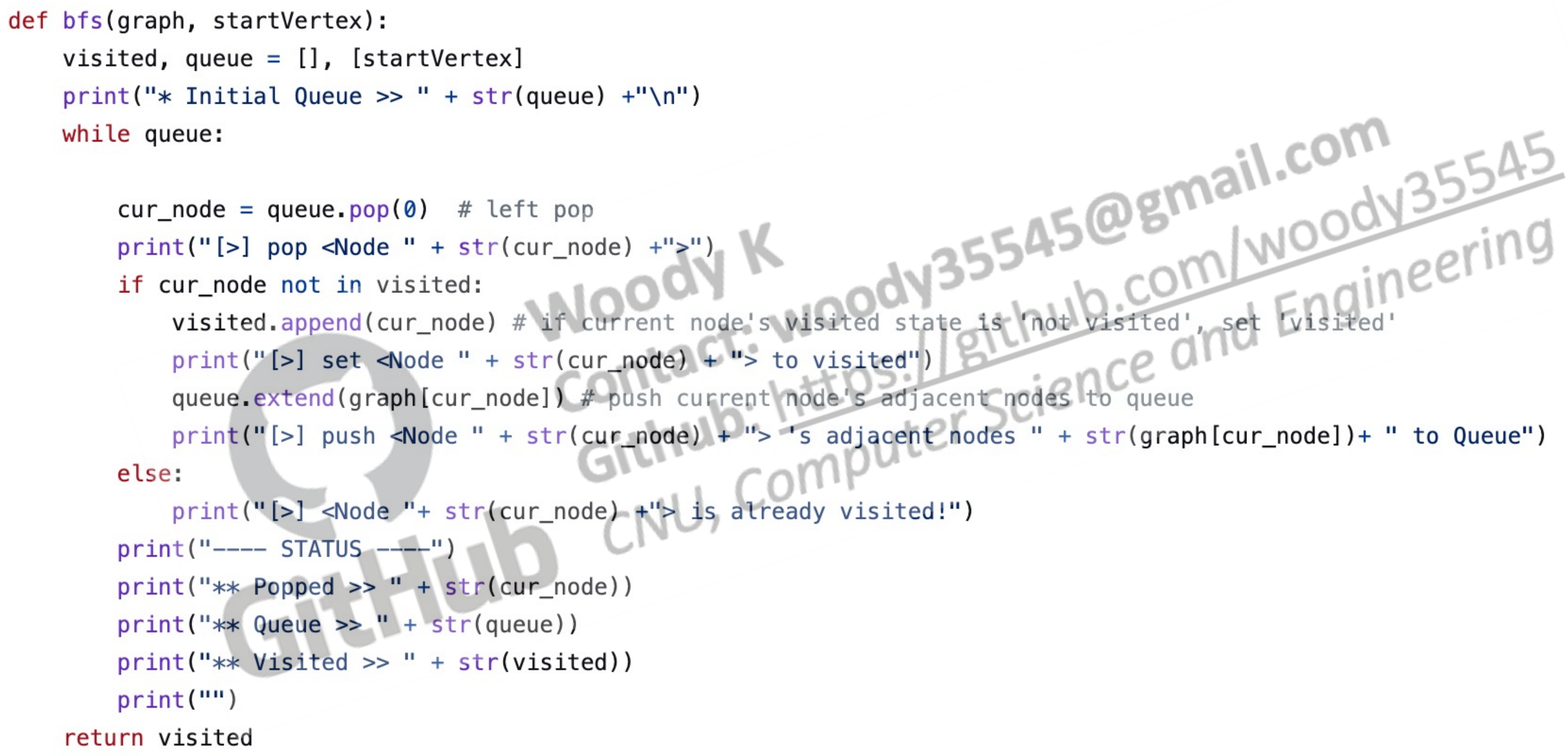


```
def bfs(graph, start_node):

    visited, queue = [], [start_node]

    while queue:
        dequeued_node = queue.pop(0)  # dequeue

        if dequeued_node not in visited:

            # if current dequeued node's visited state is 'not visited', set 'visited'
            visited.append(dequeued_node)
            # enqueue current dequeued node's adjacent nodes
            queue.extend(graph[dequeued_node])

    return visited
```

BFS Python Code

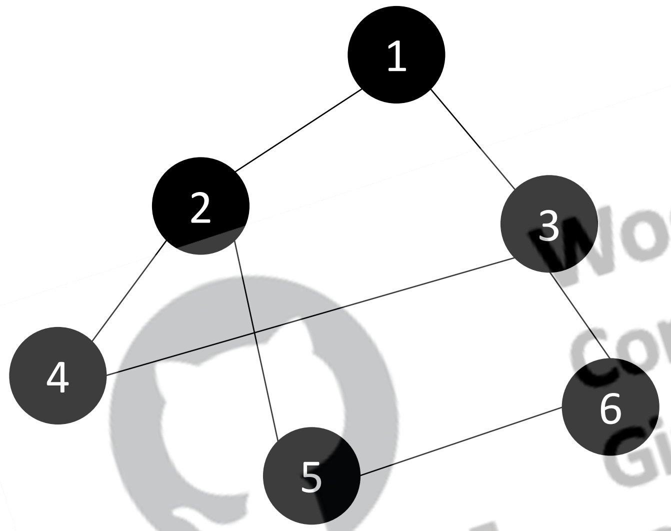
```
def bfs(graph, start_node):

    visited, queue = [], [start_node]

    while queue:

        dequeued_node = queue.pop(0)  # dequeue
        if dequeued_node not in visited:
            visited.append(dequeued_node) # if current dequeued node's visited state is 'not visited', set 'visited'
            queue.extend(graph[dequeued_node]) # enqueue current dequeued node's adjacent nodes

    return visited
```

BFS Python Code – 과정 확인을 위해 출력문 추가한 코드



```python
def bfs(graph, startVertex):
    visited, queue = [], [startVertex]
    print("* Initial Queue >> " + str(queue) +"\n")
    while queue:

        cur_node = queue.pop(0)  # left pop
        print("[>] pop <Node " + str(cur_node) +">")
        if cur_node not in visited:
            visited.append(cur_node) # if current node's visited state is 'not visited', set 'visited'
            print("[>] set <Node " + str(cur_node) + "> to visited")
            queue.extend(graph[cur_node]) # push current node's adjacent nodes to queue
            print("[>] push <Node " + str(cur_node) + "> 's adjacent nodes " + str(graph[cur_node])+ " to Queue")
        else:
            print("[>] <Node "+ str(cur_node) +"> is already visited!")
        print("---- STATUS ----")
        print("** Popped >> " + str(cur_node))
        print("** Queue >> " + str(queue))
        print("** Visited >> " + str(visited))
        print("")
    return visited
```

테스트에 사용한 Graph

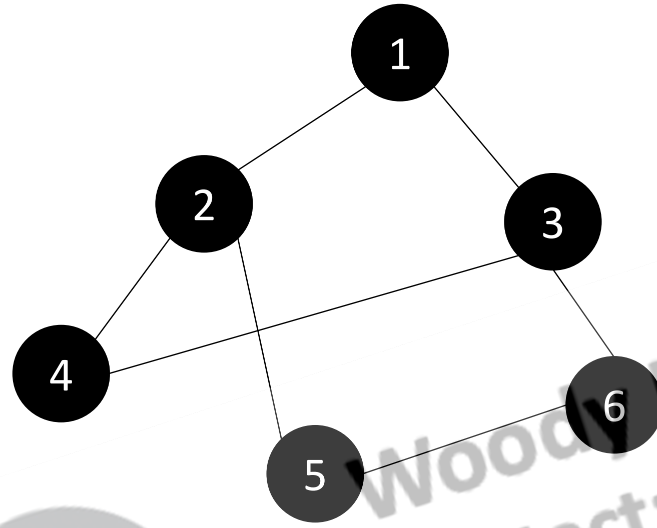


<테스트에 사용한 그래프>

```
test_graph = [[],
              [2,3],
              [1,4,5],
              [1,4,6],
              [2,3],
              [2,6],
              [3,5]
              ]
```

<인접리스트 방식으로 표현한 그래프>

< Start **BFS** with Node ”1” >



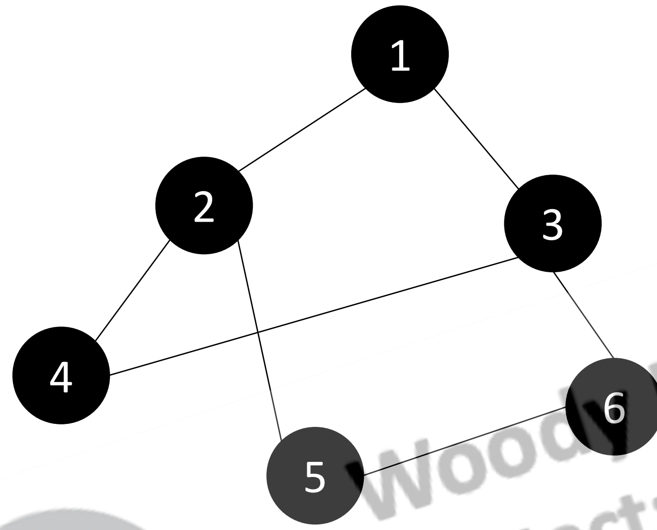
Popped

Queue

Visited

< Start **BFS** with Node ”1” >

> Push ‘Start Node(1)’ to Queue



Popped

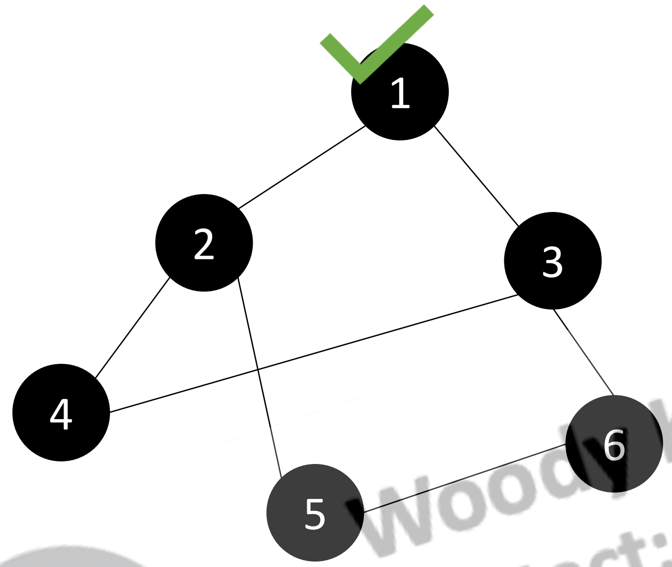
Queue

● 1									

```
* Initial Queue >> [1]
```

Visited

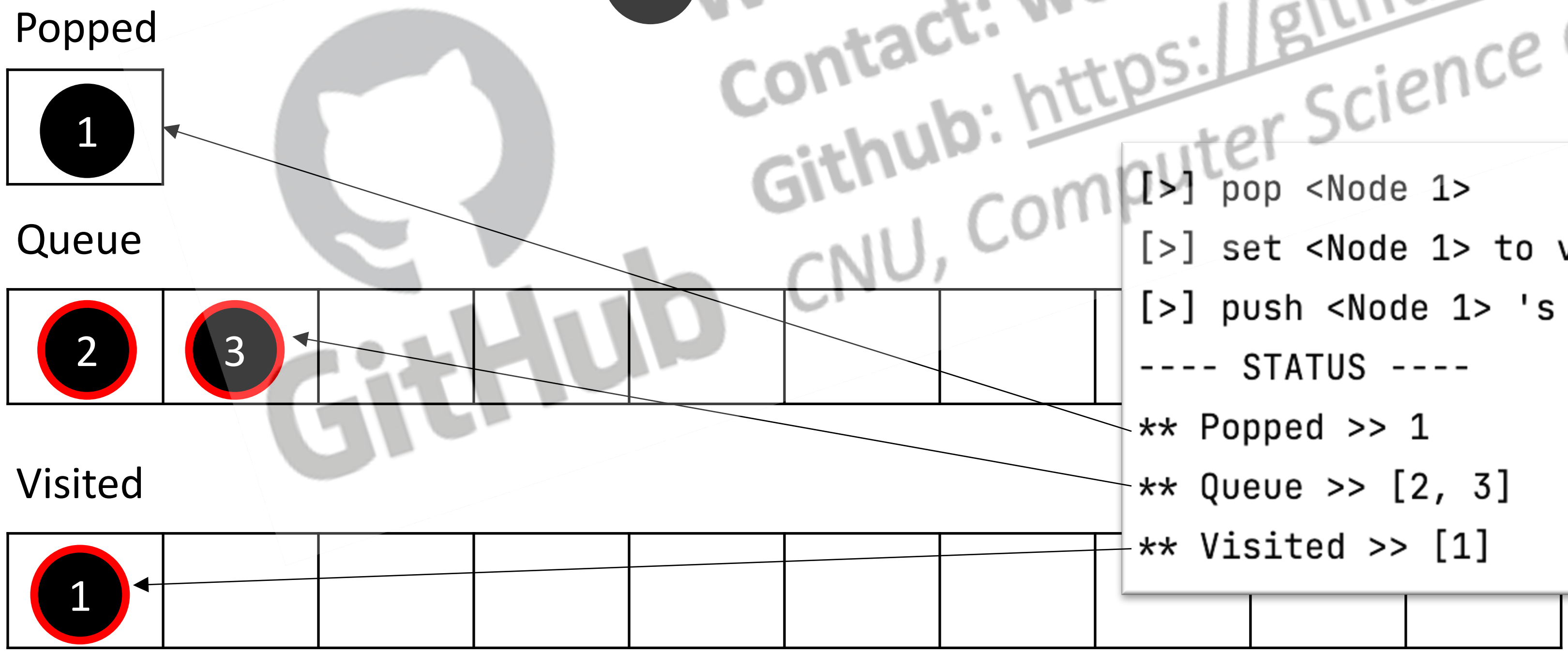
< Start **BFS** with Node ”1” >

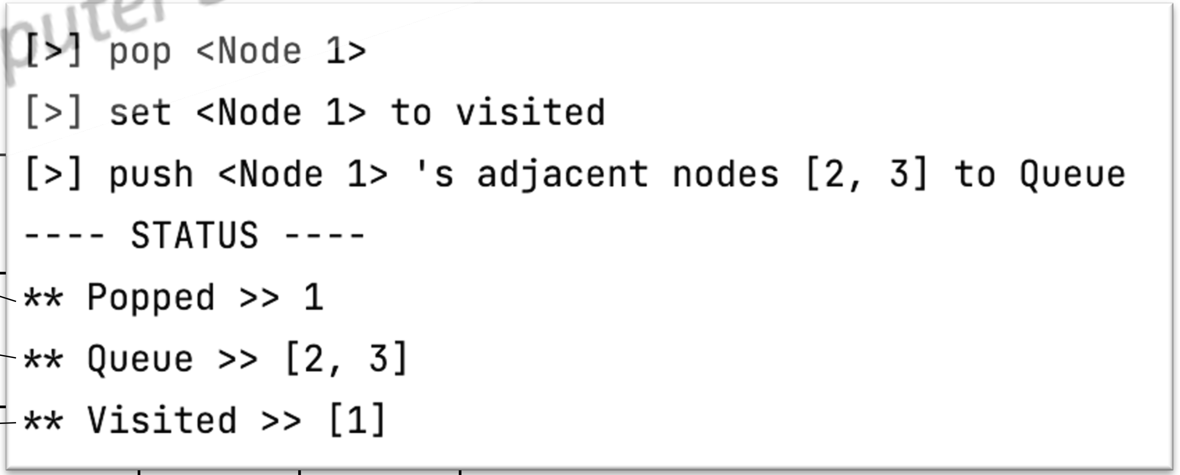


pop Queue

만약 pop 된 노드가 visited 처리가 되어있지 않을 경우
-> visited 처리

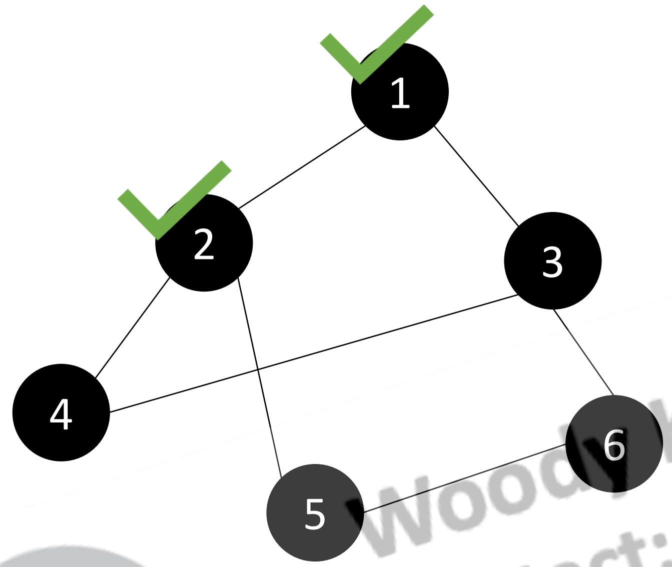
Pop된 노드의 인접 노드(2,3)를 Queue에 추가





```
[>] pop <Node 1>
[>] set <Node 1> to visited
[>] push <Node 1> 's adjacent nodes [2, 3] to Queue
---- STATUS ----
** Popped >> 1
** Queue >> [2, 3]
** Visited >> [1]
```

< Start **BFS** with Node ”1” >



pop Queue (popped = 2)

만약 pop 된 노드가 visited 처리가 되어있지 않을 경우
-> pop 된 노드(2)를 visited 처리

pop된 노드의 인접 노드(1,4,5)를 Queue에 추가

Popped

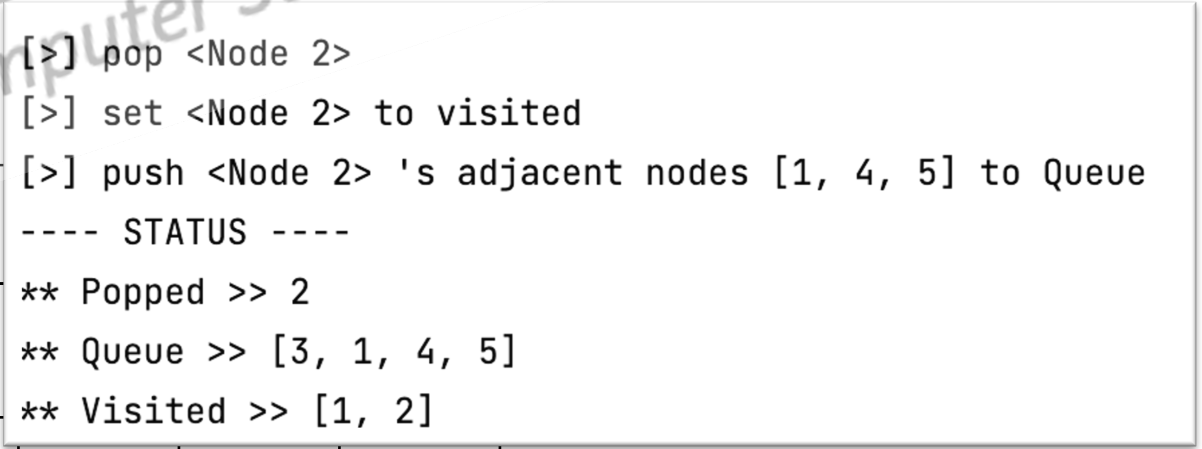
● 2

Queue

3	1	4	5			

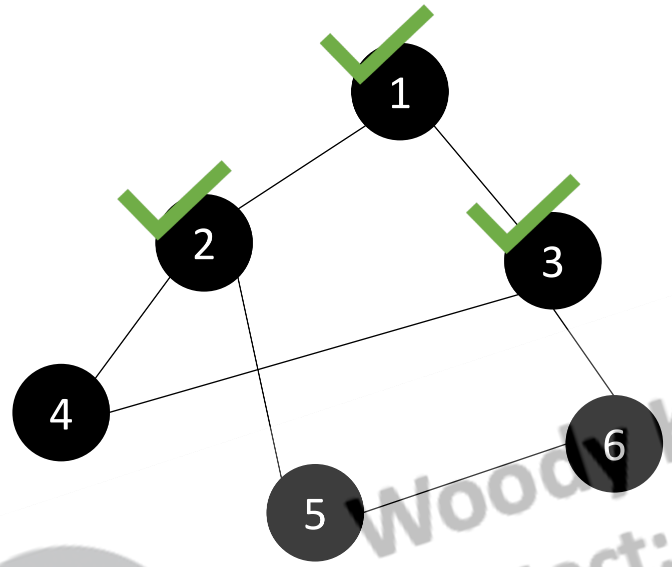
Visited

1	2								



```
[>] pop <Node 2>
[>] set <Node 2> to visited
[>] push <Node 2> 's adjacent nodes [1, 4, 5] to Queue
---- STATUS ----
** Popped >> 2
** Queue >> [3, 1, 4, 5]
** Visited >> [1, 2]
```

< Start **BFS** with Node ”1” >



pop Queue (popped = 3)

만약 pop 된 노드가 visited 처리가 되어있지 않을 경우
-> pop 된 노드(3)를 visited 처리

pop된 노드의 인접 노드(1,4,6)를 Queue에 추가

Popped

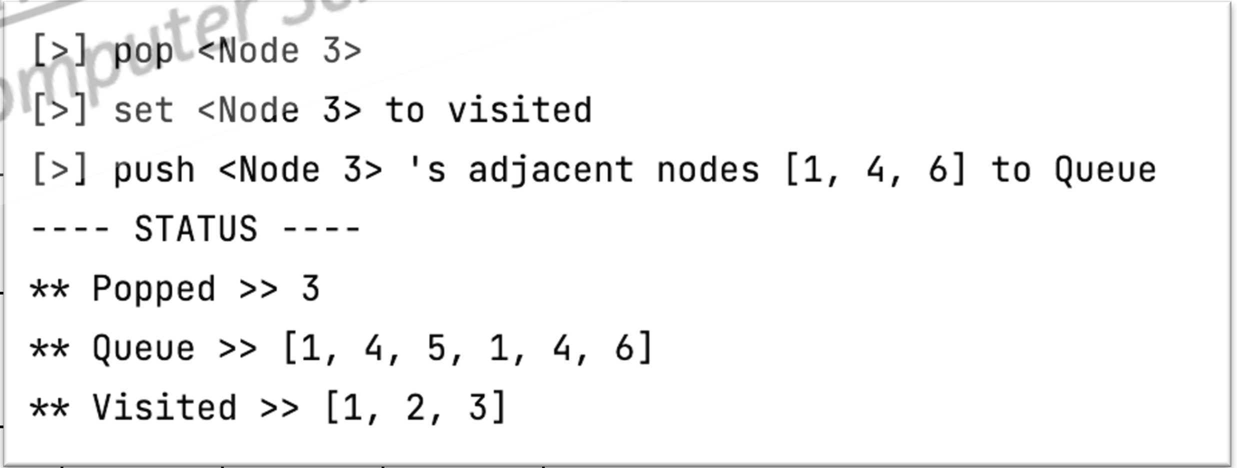
3

Queue

1	4	5	1	4	6	

Visited

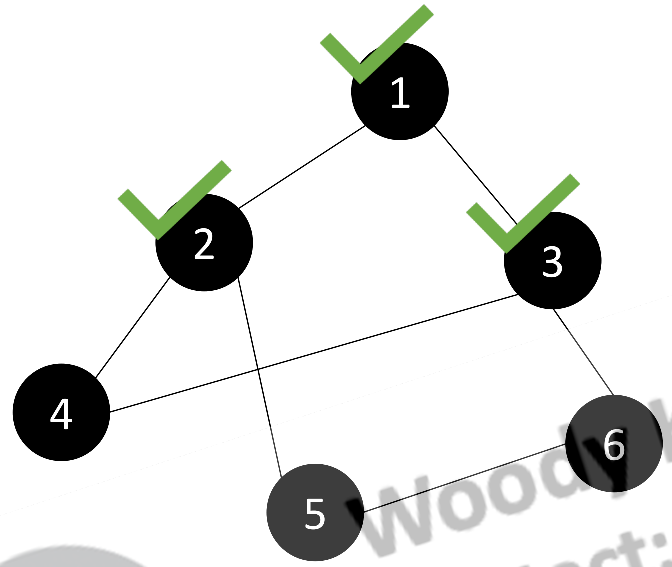
1	2	3							



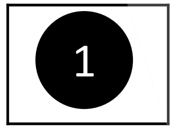
```
[>] pop <Node 3>
[>] set <Node 3> to visited
[>] push <Node 3> 's adjacent nodes [1, 4, 6] to Queue
---- STATUS ----
** Popped >> 3
** Queue >> [1, 4, 5, 1, 4, 6]
** Visited >> [1, 2, 3]
```

< Start **BFS** with Node ”1” >

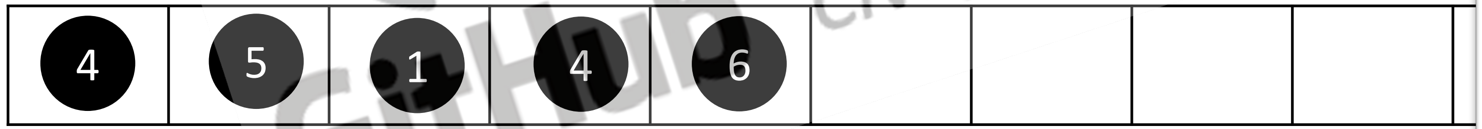
Node 1은 이미 방문한 상태이므로 아무것도 하지 않음



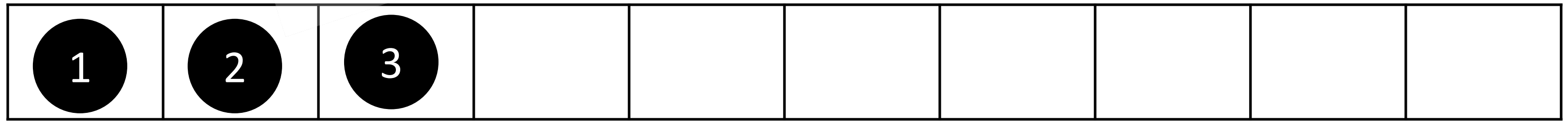
Popped



Queue

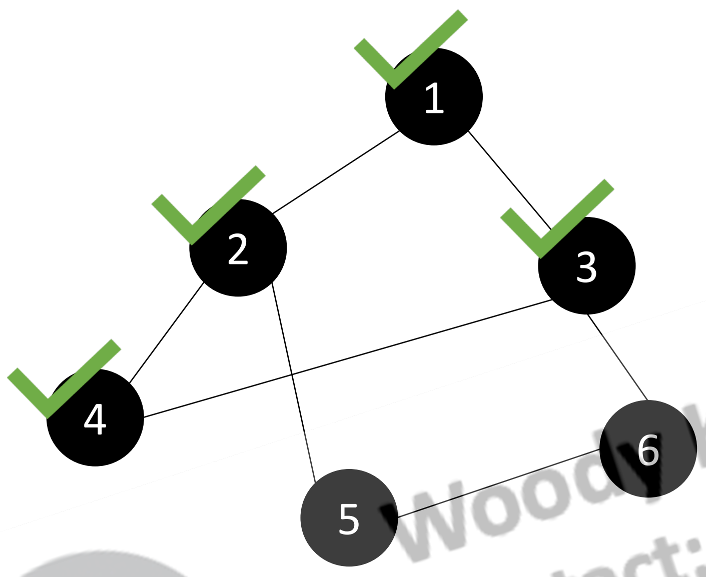


Visited



```
[>] pop <Node 1>
[>] <Node 1> is already visited!
---- STATUS ----
** Popped >> 1
** Queue >> [4, 5, 1, 4, 6]
** Visited >> [1, 2, 3]
```

< Start **BFS** with Node ”1” >

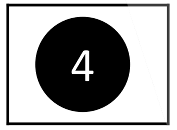


pop Queue (popped = 4)

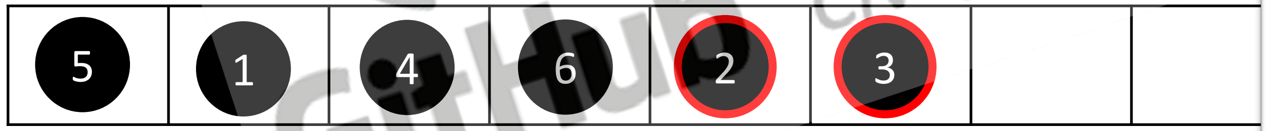
만약 pop 된 노드가 visited 처리가 되어있지 않을 경우
-> pop 된 노드(4)를 visited 처리

pop된 노드의 인접 노드(2,3)를 Queue에 추가

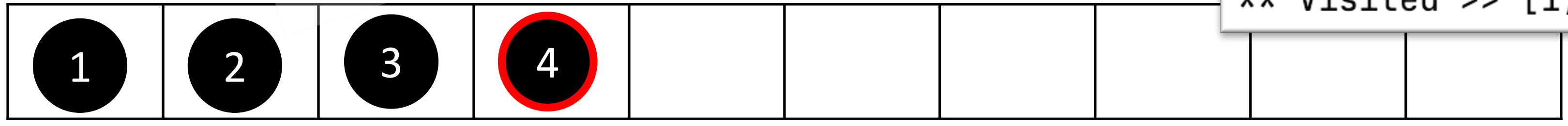
Popped

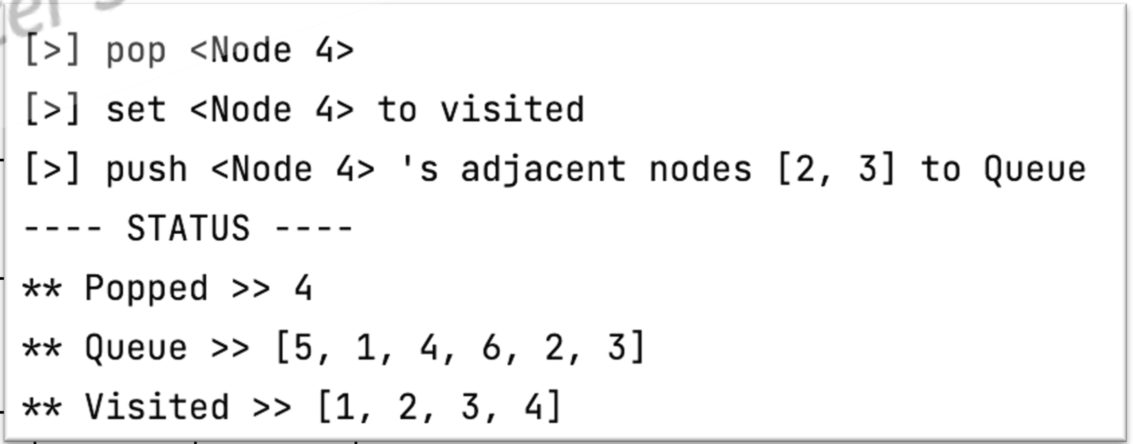


Queue



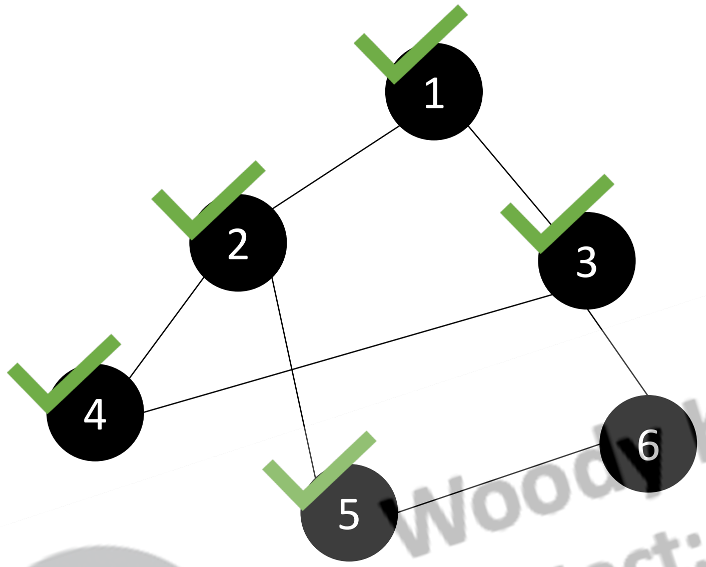
Visited





```
[>] pop <Node 4>
[>] set <Node 4> to visited
[>] push <Node 4> 's adjacent nodes [2, 3] to Queue
---- STATUS ----
** Popped >> 4
** Queue >> [5, 1, 4, 6, 2, 3]
** Visited >> [1, 2, 3, 4]
```

< Start **BFS** with Node ”1” >



pop Queue (popped = 5)

만약 pop 된 노드가 visited 처리가 되어있지 않을 경우
-> pop 된 노드(5)를 visited 처리

pop된 노드의 인접 노드(2,6)를 Queue에 추가

Popped

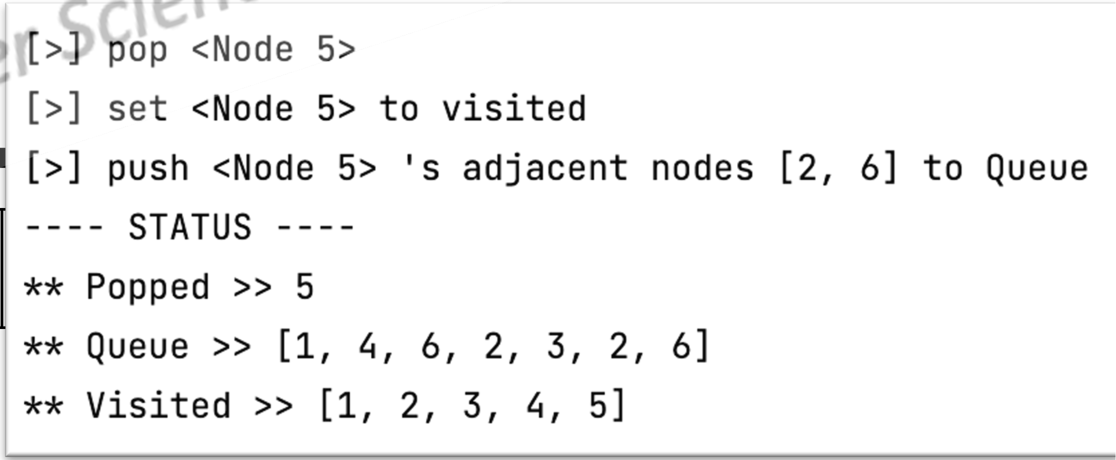
5

Queue

1	4	6	2	3	2	6	

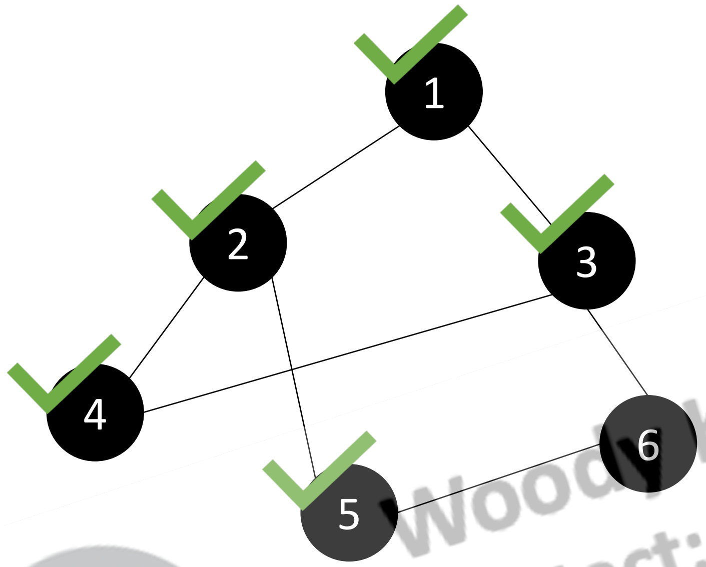
Visited

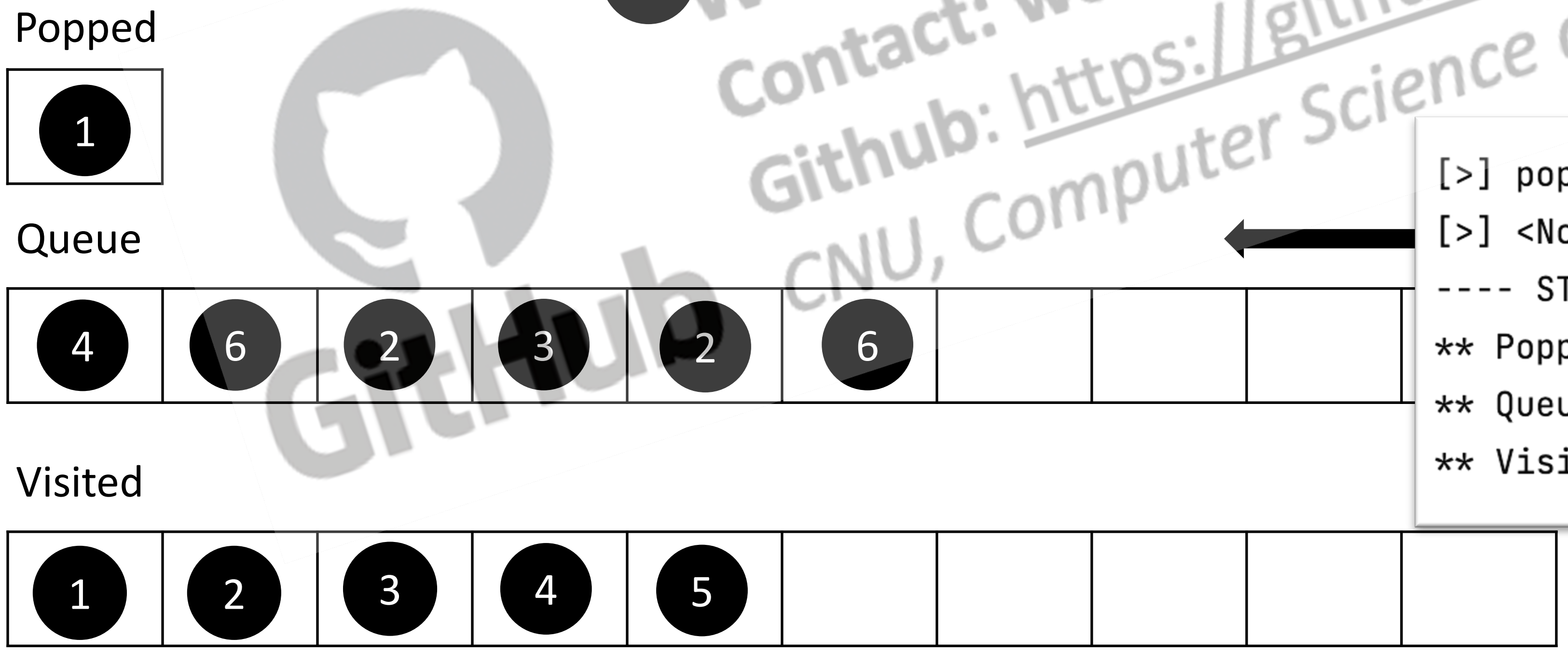
1	2	3	4	5					

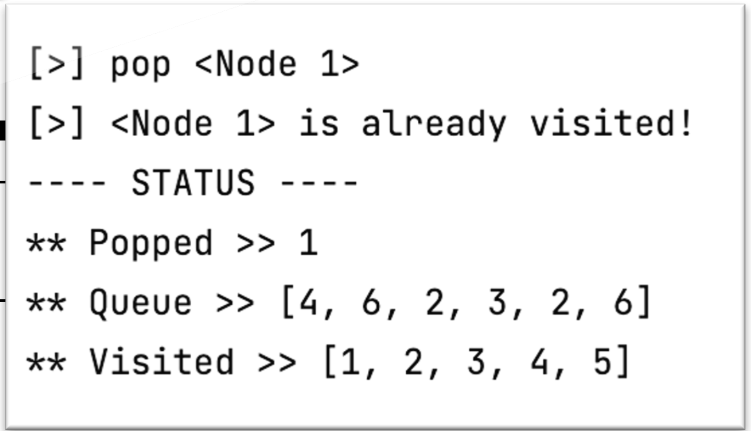

```
[>] pop <Node 5>
[>] set <Node 5> to visited
[>] push <Node 5> 's adjacent nodes [2, 6] to Queue
---- STATUS ----
** Popped >> 5
** Queue >> [1, 4, 6, 2, 3, 2, 6]
** Visited >> [1, 2, 3, 4, 5]
```

< Start **BFS** with Node ”1” >

Node 1은 이미 방문한 상태이므로 아무것도 하지 않음



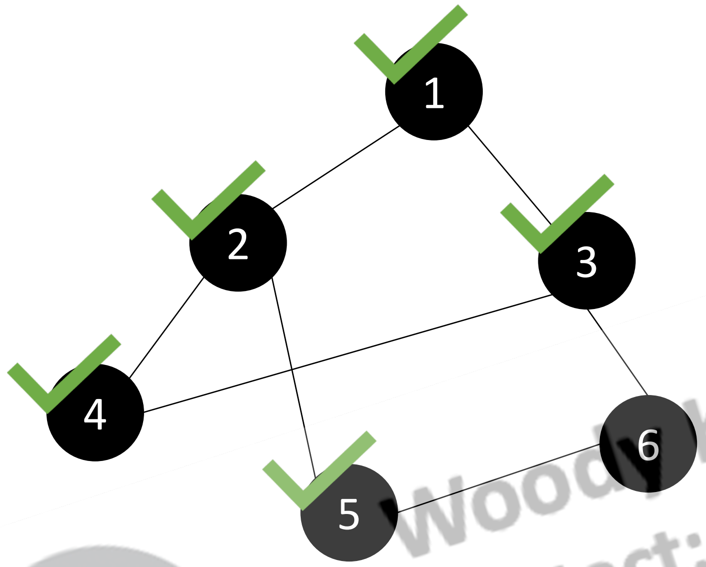




```
[>] pop <Node 1>
[>] <Node 1> is already visited!
---- STATUS ----
** Popped >> 1
** Queue >> [4, 6, 2, 3, 2, 6]
** Visited >> [1, 2, 3, 4, 5]
```

< Start **BFS** with Node ”1” >

Node 4는 이미 방문한 상태이므로 아무것도 하지 않음



Popped

4

Queue

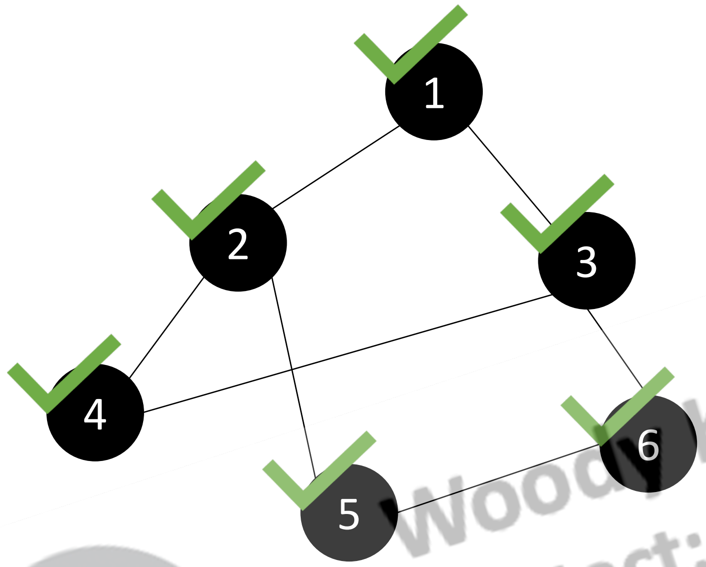
6	2	3	2	6				

Visited

1	2	3	4	5					

```
[>] pop <Node 4>
[>] <Node 4> is already visited!
---- STATUS ----
** Popped >> 4
** Queue >> [6, 2, 3, 2, 6]
** Visited >> [1, 2, 3, 4, 5]
```

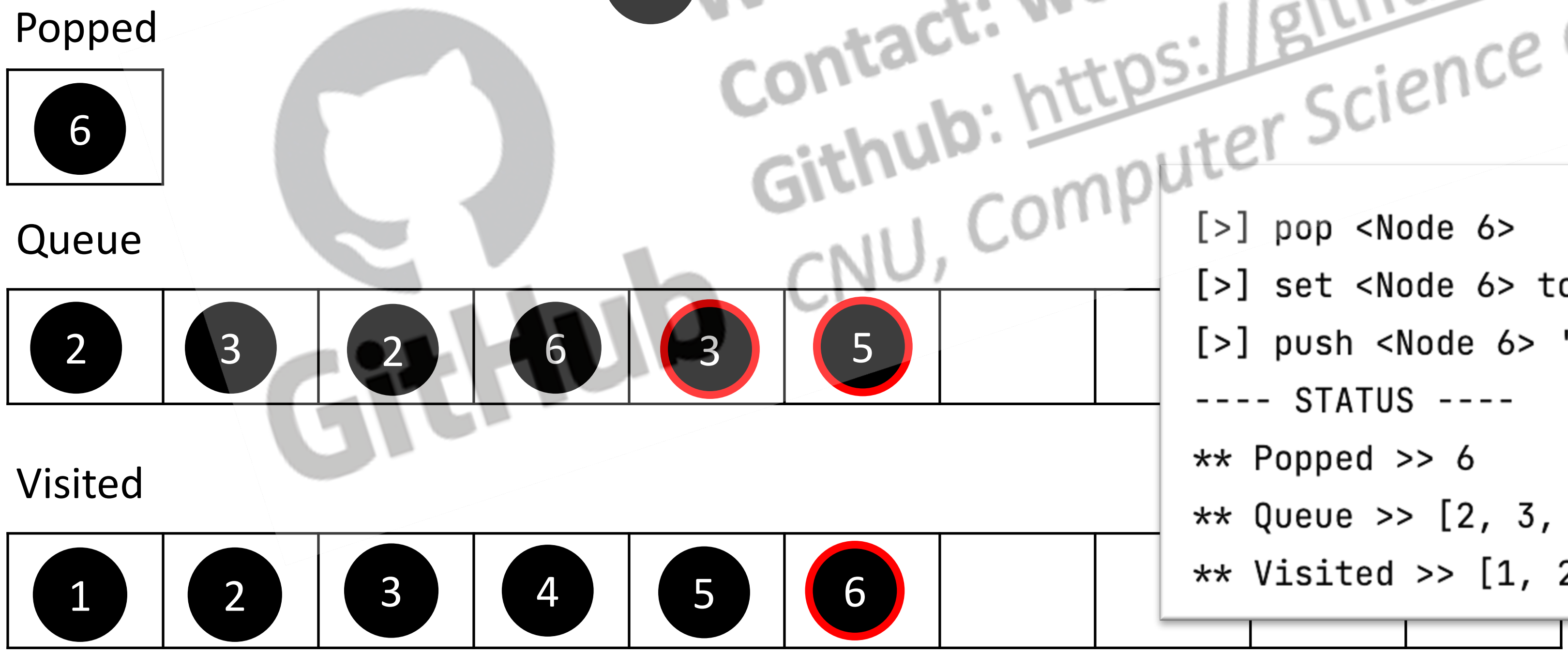
< Start **BFS** with Node ”1” >

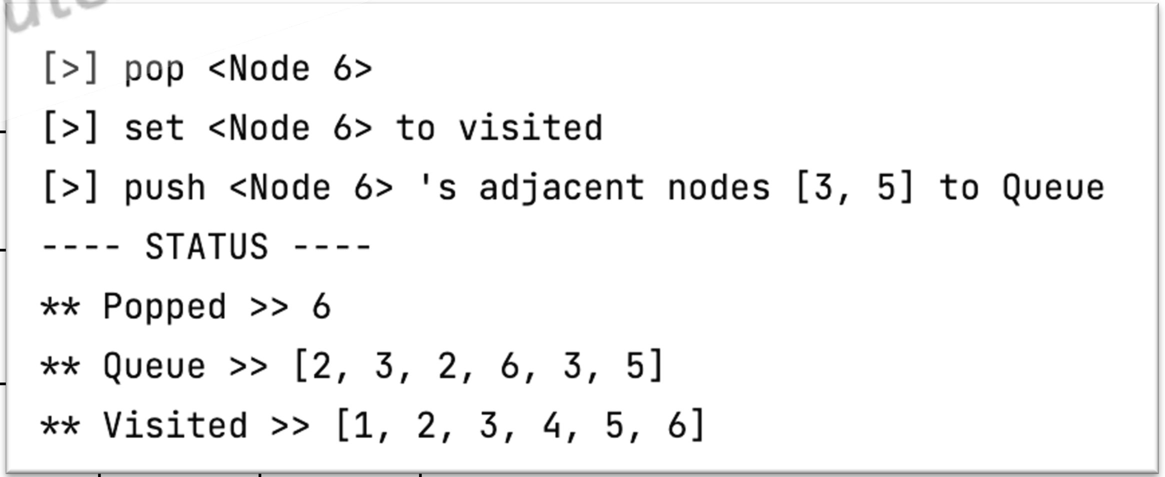


pop Queue (popped = 6)

만약 pop 된 노드가 visited 처리가 되어있지 않을 경우
-> pop 된 노드(6)를 visited 처리

pop된 노드의 인접 노드(3,5)를 Queue에 추가

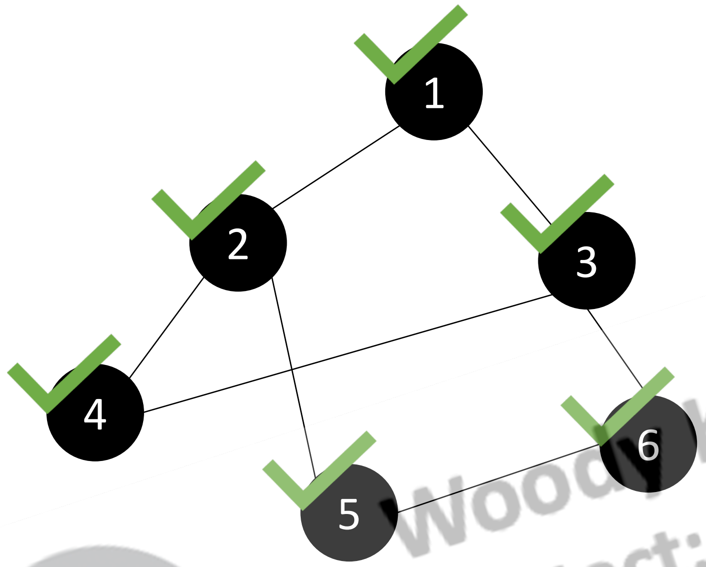




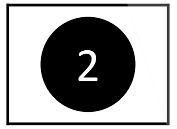
```
[>] pop <Node 6>
[>] set <Node 6> to visited
[>] push <Node 6> 's adjacent nodes [3, 5] to Queue
---- STATUS ----
** Popped >> 6
** Queue >> [2, 3, 2, 6, 3, 5]
** Visited >> [1, 2, 3, 4, 5, 6]
```

< Start **BFS** with Node ”1” >

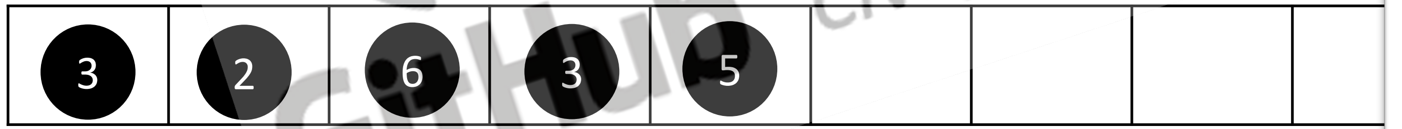
Node 2은 이미 방문한 상태이므로 아무것도 하지 않음



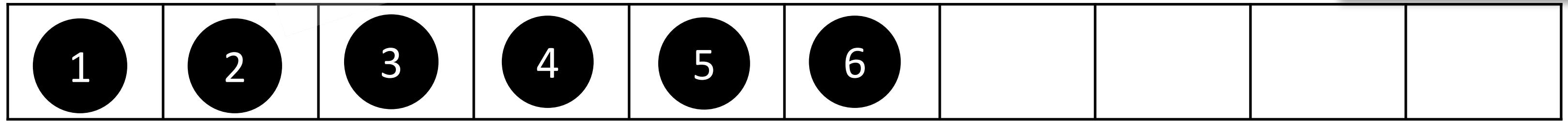
Popped

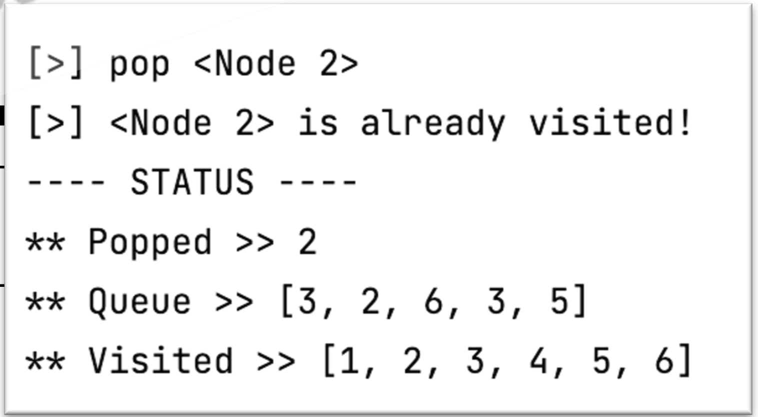


Queue



Visited

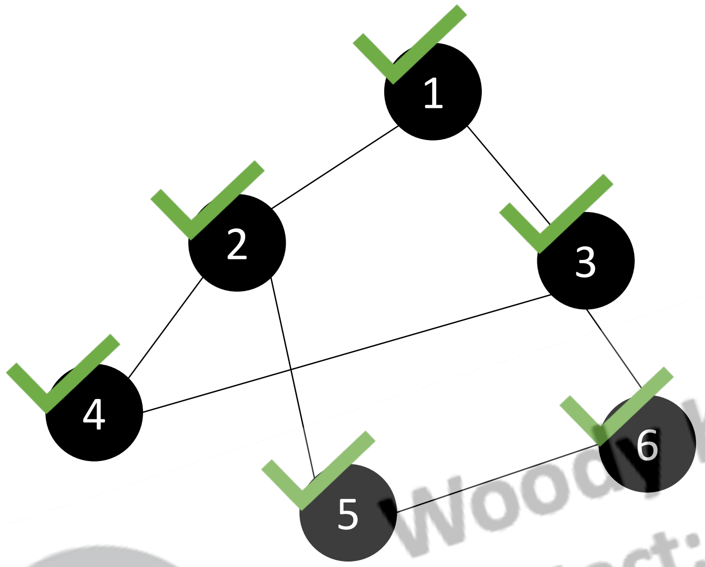




```
[>] pop <Node 2>
[>] <Node 2> is already visited!
---- STATUS ----
** Popped >> 2
** Queue >> [3, 2, 6, 3, 5]
** Visited >> [1, 2, 3, 4, 5, 6]
```

< Start **BFS** with Node "1" >

Node 3은 이미 방문한 상태이므로 아무것도 하지 않음



Popped

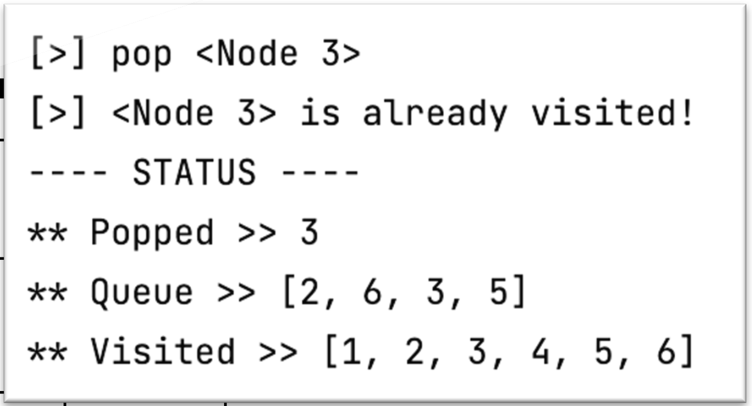
3

Queue

2	6	3	5						

Visited

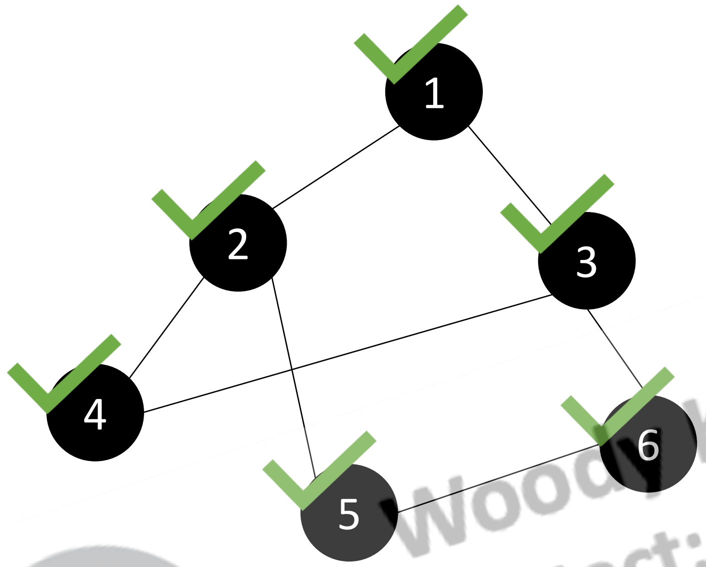
1	2	3	4	5	6				



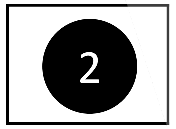
```
[>] pop <Node 3>
[>] <Node 3> is already visited!
---- STATUS ----
** Popped >> 3
** Queue >> [2, 6, 3, 5]
** Visited >> [1, 2, 3, 4, 5, 6]
```

< Start **BFS** with Node "1" >

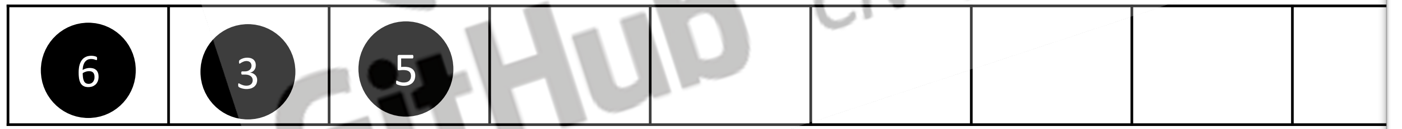
Node 2은 이미 방문한 상태이므로 아무것도 하지 않음



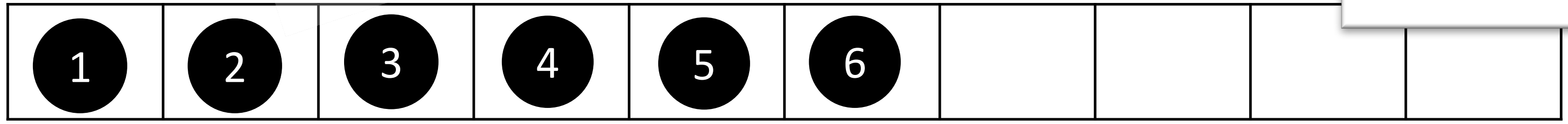
Popped

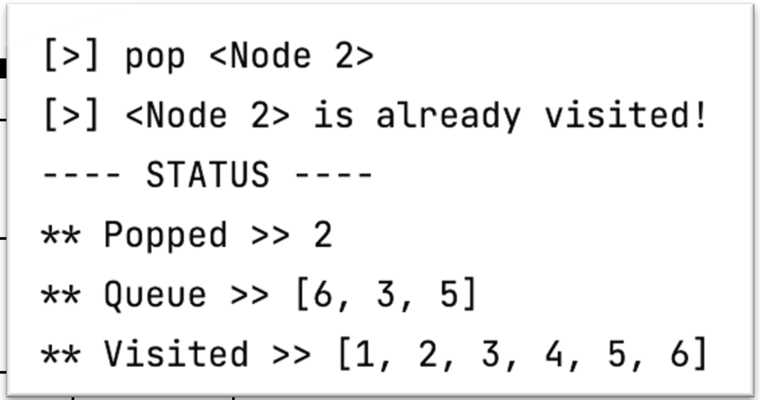


Queue



Visited

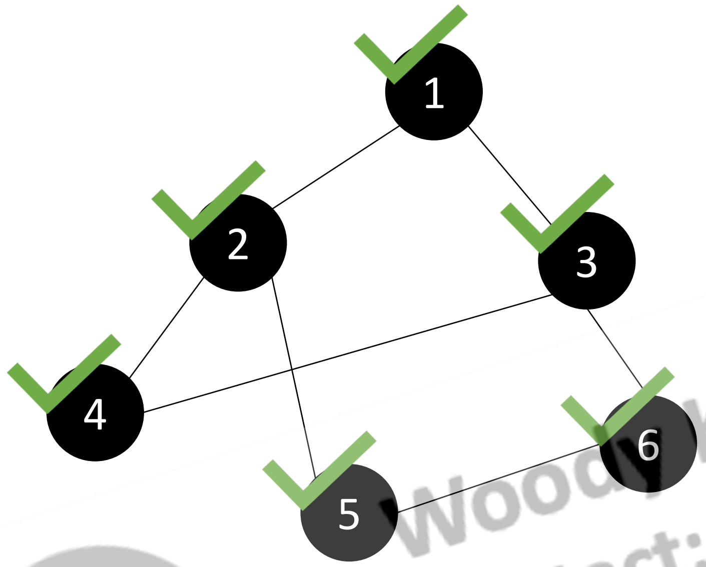




```
[>] pop <Node 2>
[>] <Node 2> is already visited!
---- STATUS ----
** Popped >> 2
** Queue >> [6, 3, 5]
** Visited >> [1, 2, 3, 4, 5, 6]
```

< Start **BFS** with Node ”1” >

Node 6은 이미 방문한 상태이므로 아무것도 하지 않음



Popped

6

Queue

3	5							

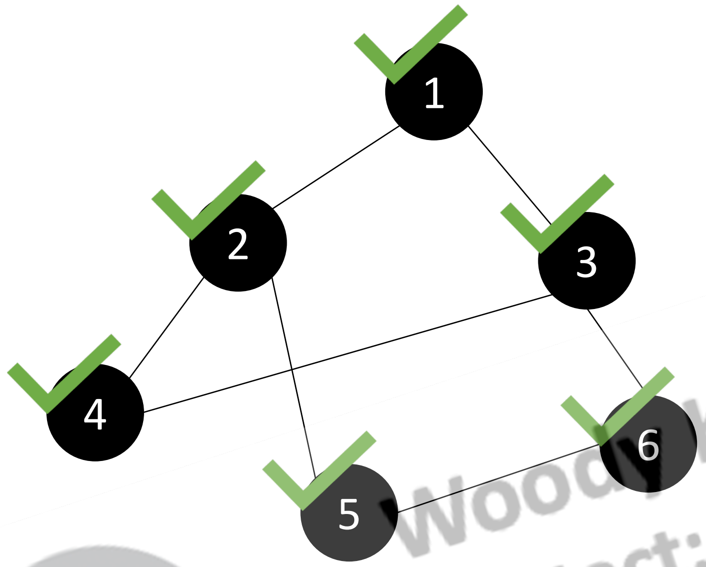
Visited

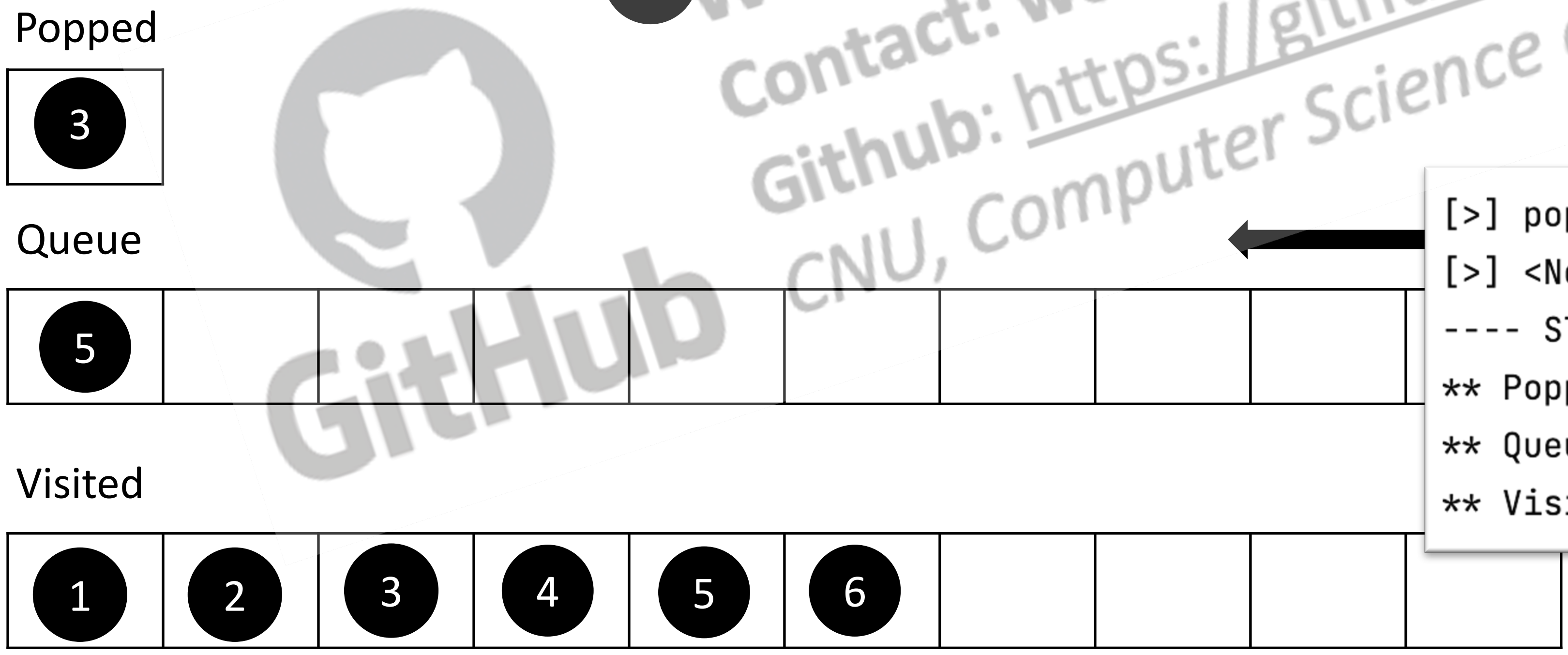
1	2	3	4	5	6			

```
[>] pop <Node 6>
[>] <Node 6> is already visited!
---- STATUS ----
** Popped >> 6
** Queue >> [3, 5]
** Visited >> [1, 2, 3, 4, 5, 6]
```

< Start **BFS** with Node ”1” >

Node 3은 이미 방문한 상태이므로 아무것도 하지 않음

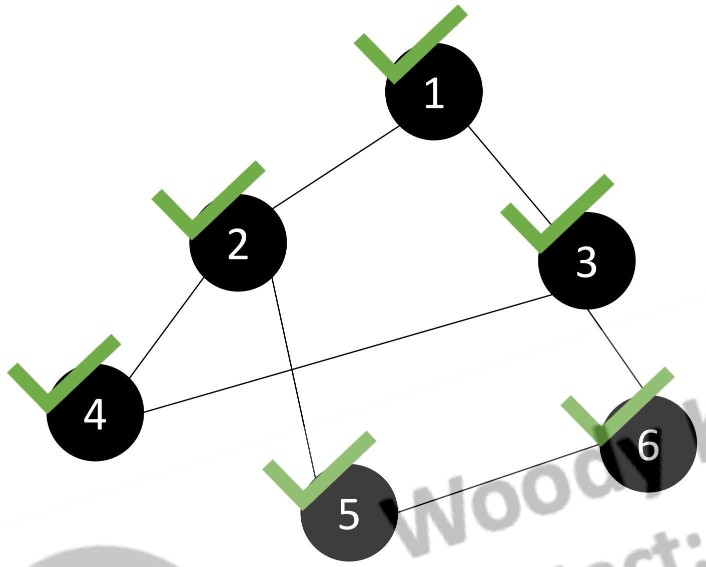


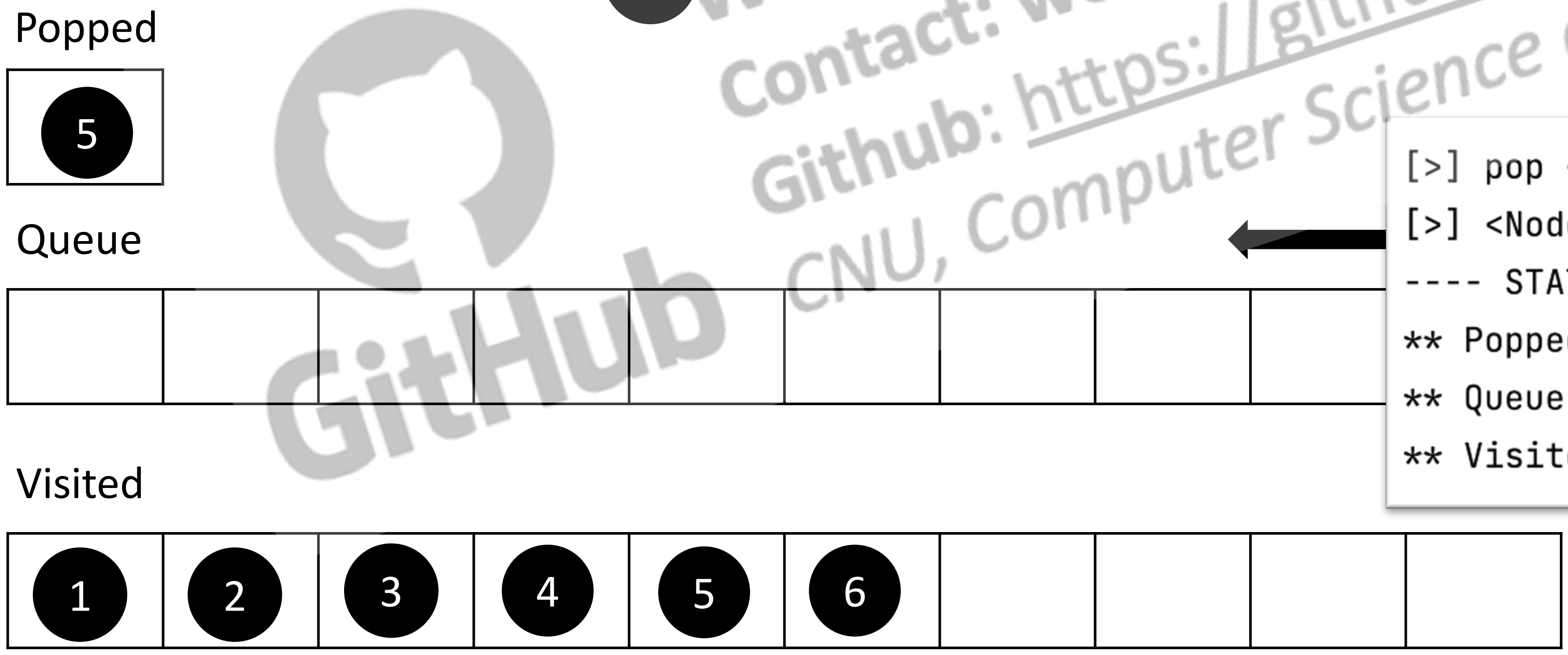


```
[>] pop <Node 3>
[>] <Node 3> is already visited!
---- STATUS ----
** Popped >> 3
** Queue >> [5]
** Visited >> [1, 2, 3, 4, 5, 6]
```

< Start **BFS** with Node ”1” >

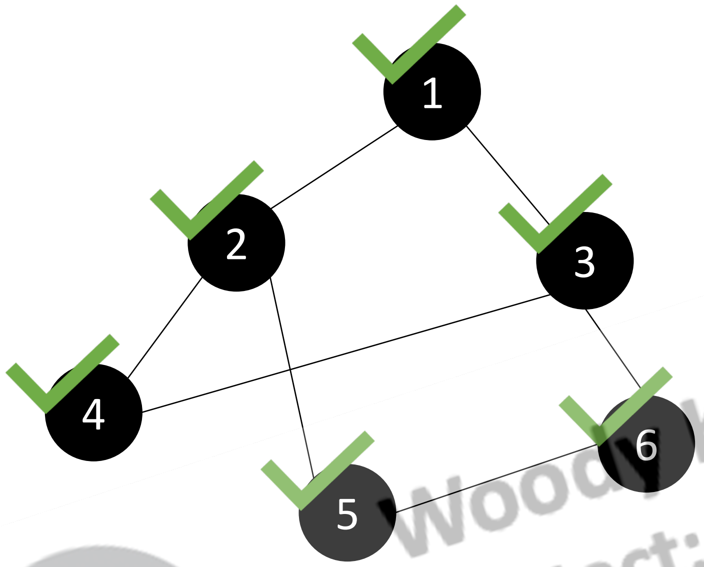
Node 5는 이미 방문한 상태이므로 아무것도 하지 않음





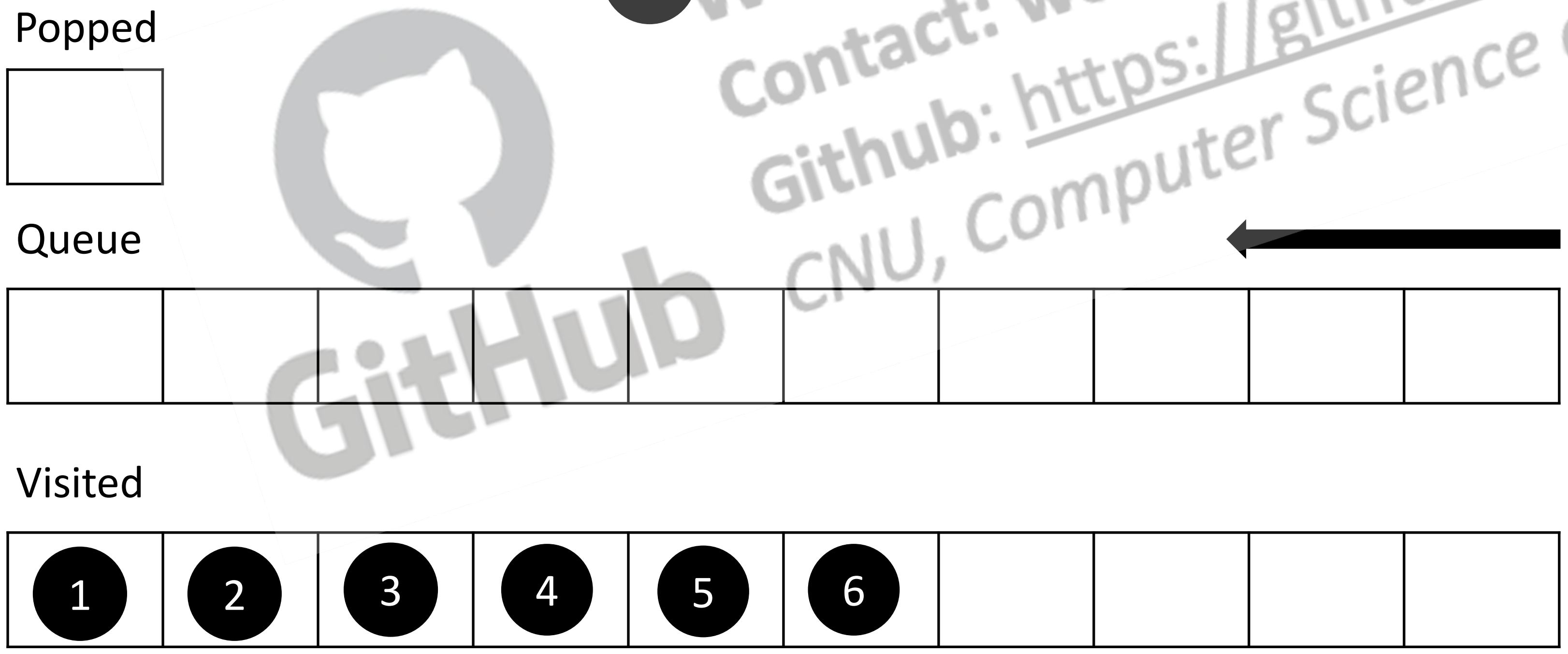
```
[>] pop <Node 5>
[>] <Node 5> is already visited!
---- STATUS ----
** Popped >> 5
** Queue >> []
** Visited >> [1, 2, 3, 4, 5, 6]
```

< Start **BFS** with Node ”1” >



Queue가 비었으므로 탐색을 종료하고 visited 리스트를 반환

BFS 순회결과: 1, 2, 3, 4, 5, 6


Popped
Queue
Visited
1
2
3
4
5
6