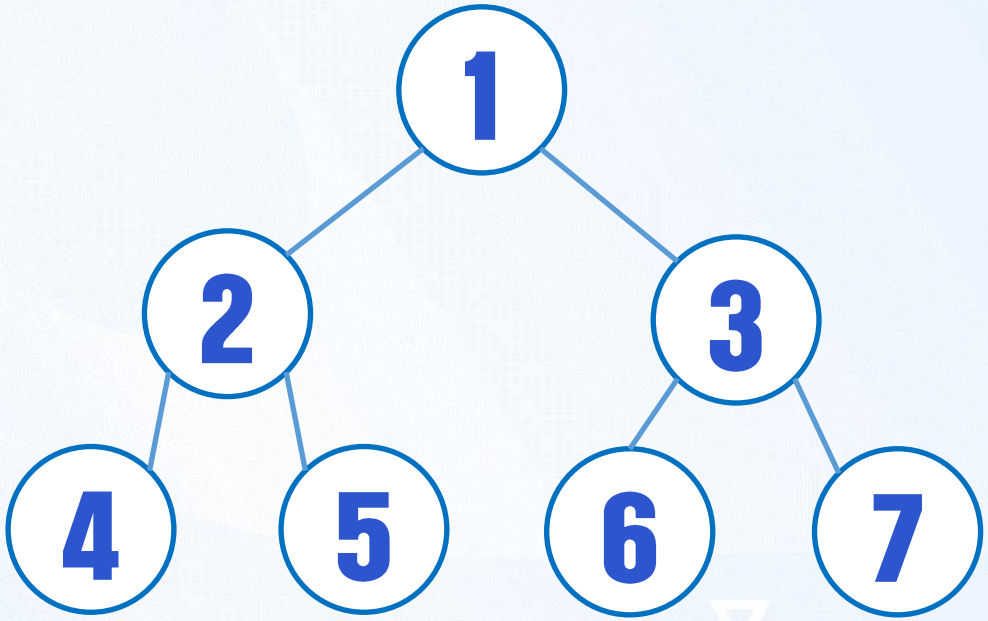
# 이진 트리

## 후위순회

# 이진트리 후위순회




- 자식(좌) → 자식(우)→ 부모 노드 순으로 순회
- 재귀 호출을 이용
- 부모가 n이라면
  - 자식(좌) = n * 2
  - 자식(우) = n * 2 + 1

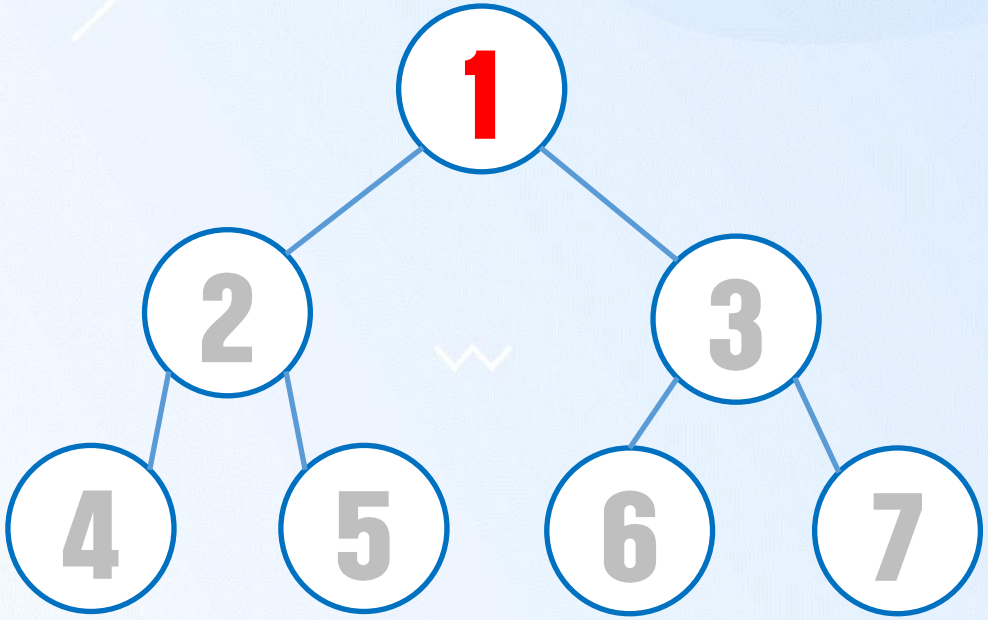
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=1

DFS(n * 2); → 2

DFS(n * 2 + 1); → 3

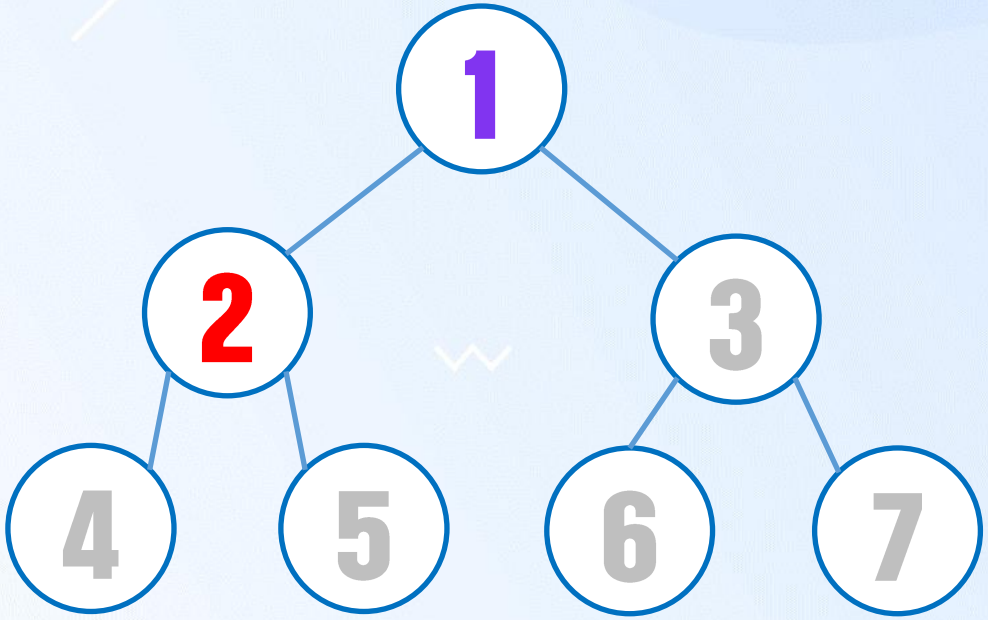




결과:

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {      n=2
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);   → 4
            DFS(n * 2 + 1); → 5
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```



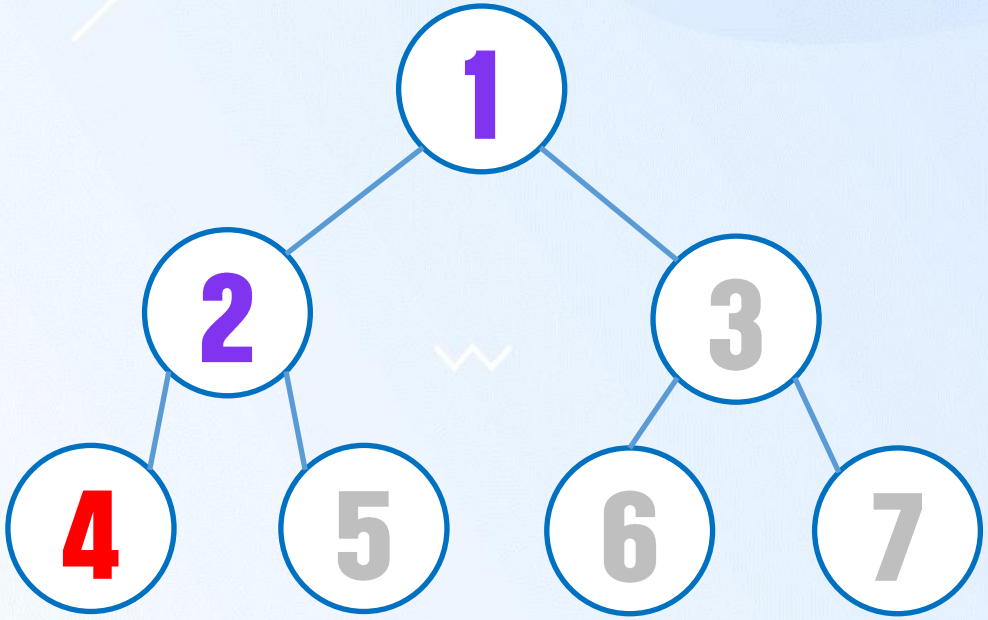


결과:

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);  → 8
            DFS(n * 2 + 1); → 9
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=4





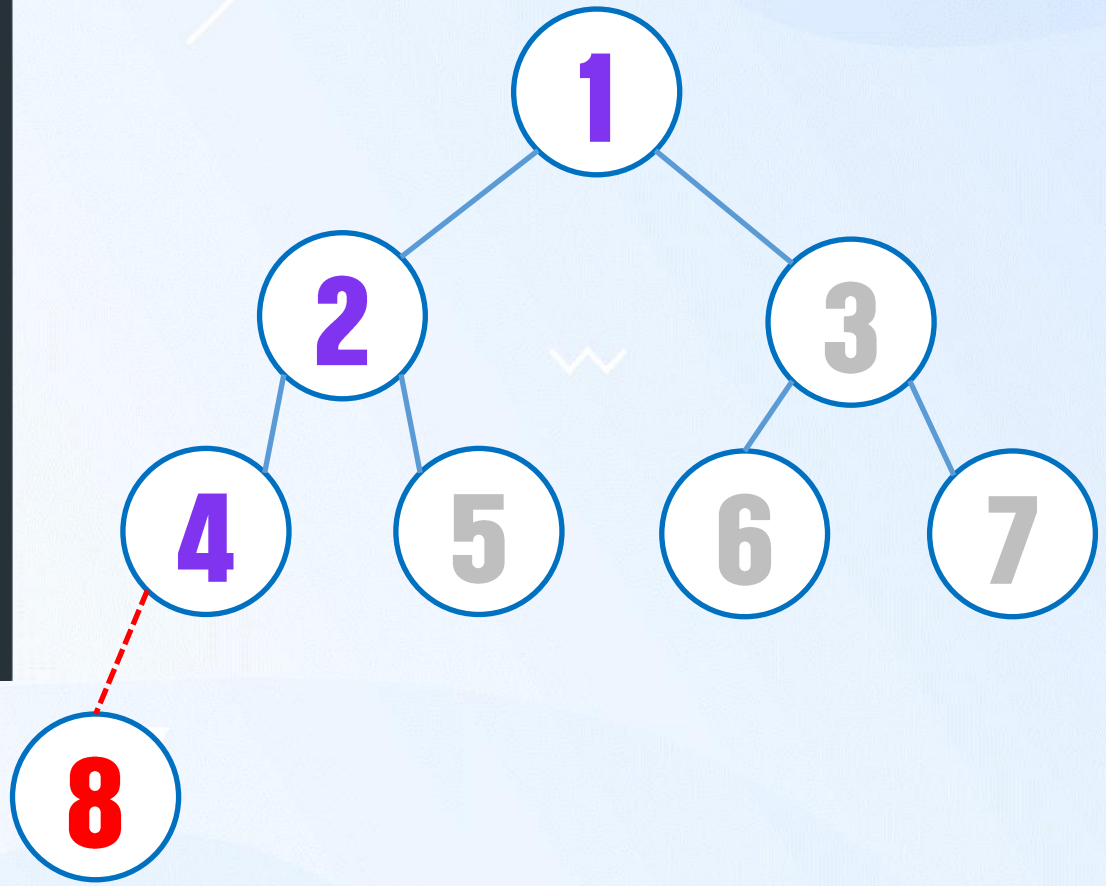
STACK

결과:

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=8





STACK

결과:

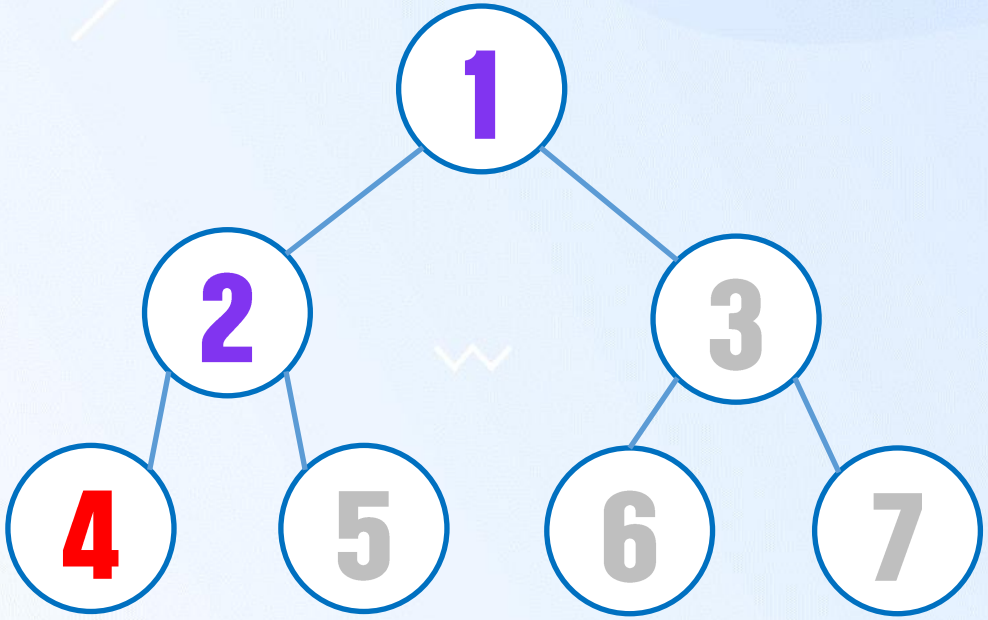
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=4

→ 8

→ 9





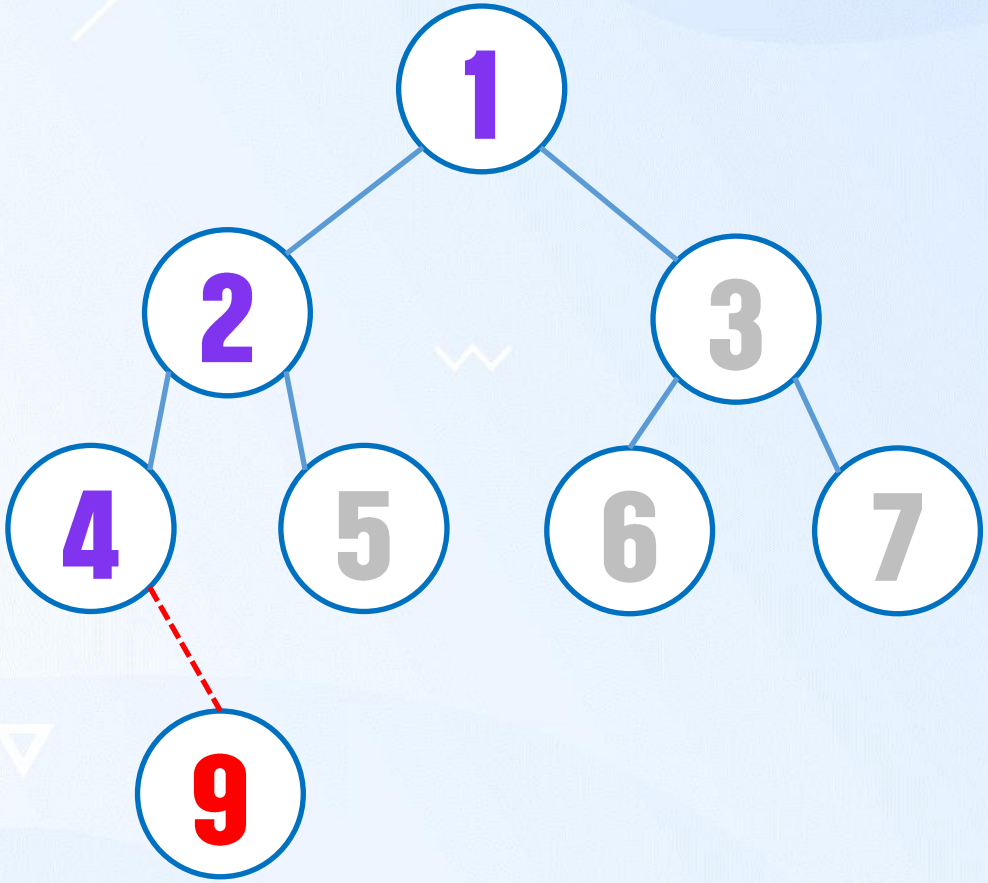
STACK

결과:

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=9



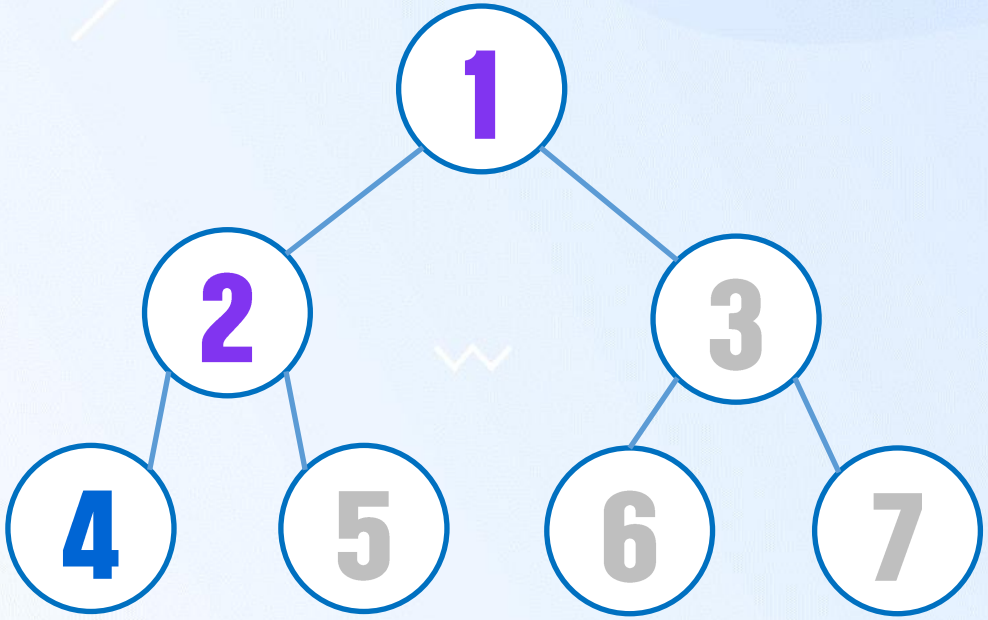


STACK

결과:

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {      n=4
        // 🏁 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);   → 8
            DFS(n * 2 + 1); → 9
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```





결과: 4

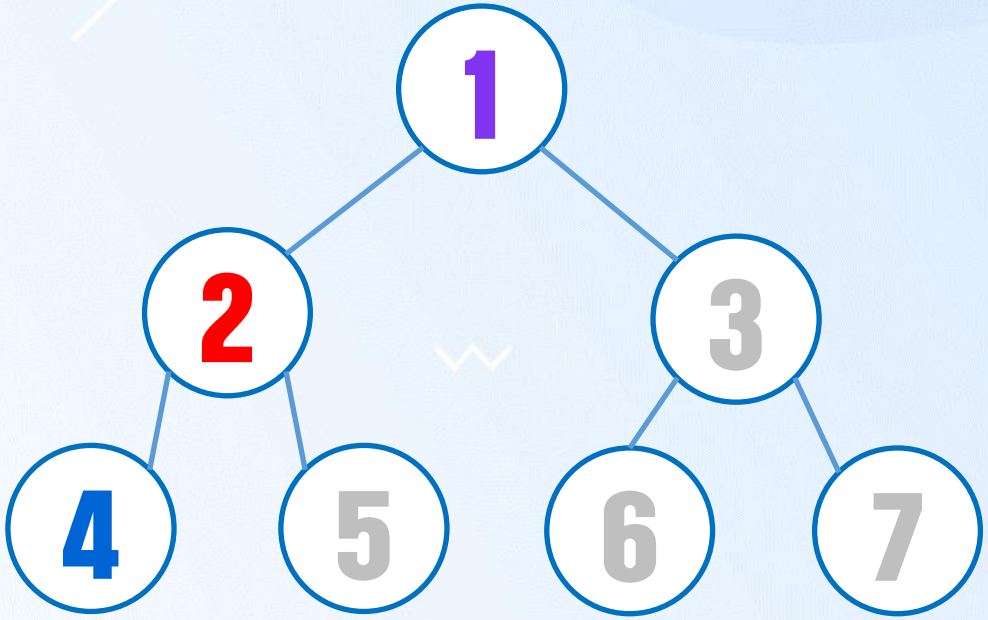
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=2

→ 4

→ 5





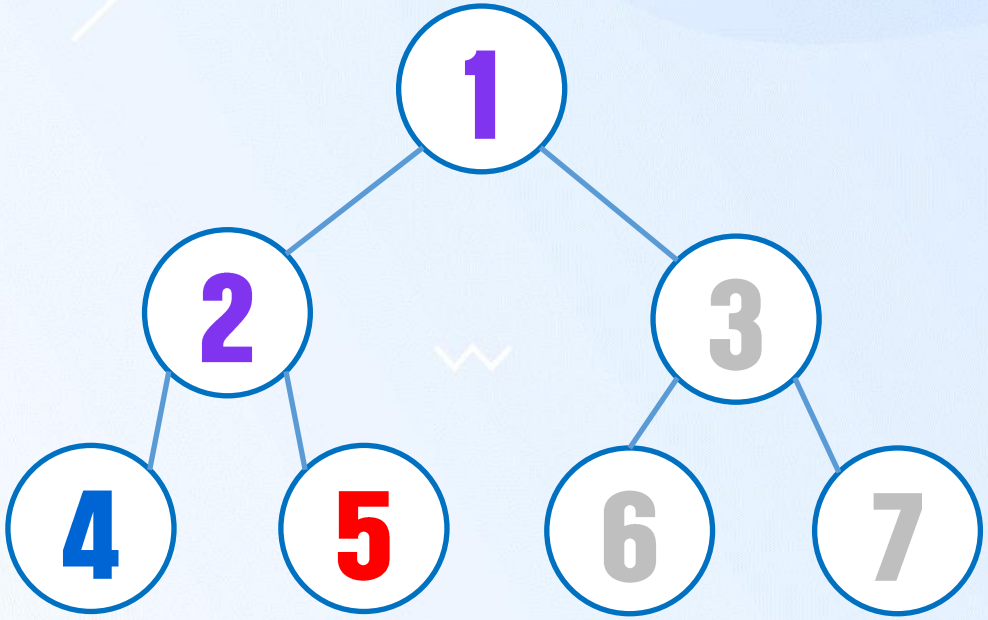
STACK

결과: 4

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);  → 10
            DFS(n * 2 + 1); → 11
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=5



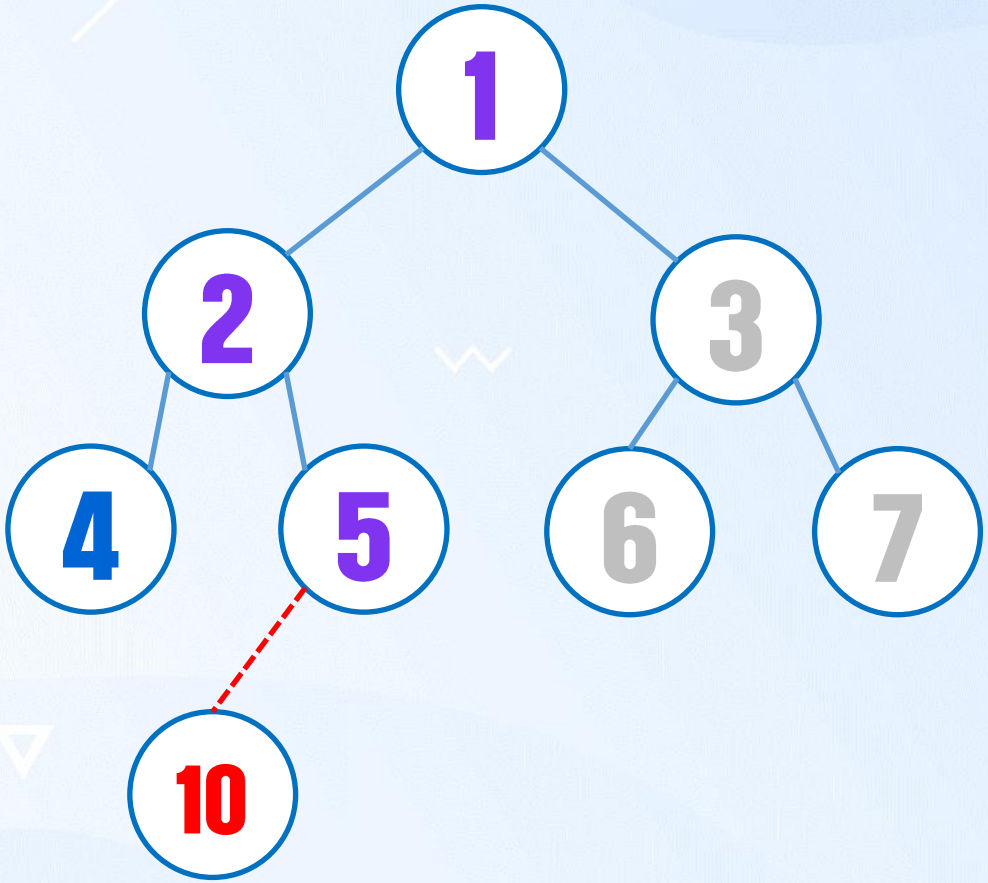


결과: 4

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=10





STACK

결과: 4

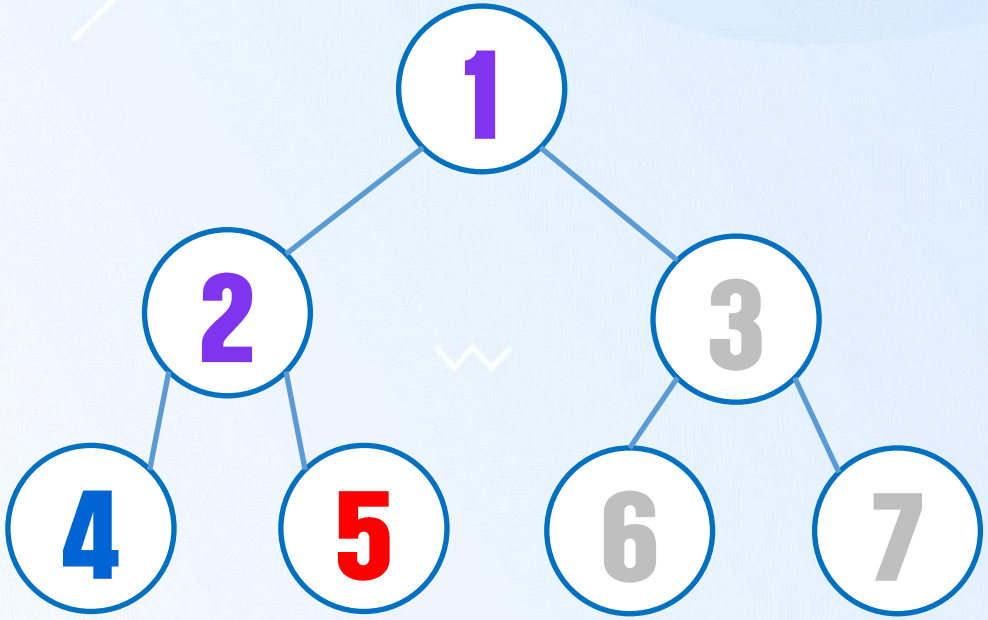
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=5

→ 10

→ 11





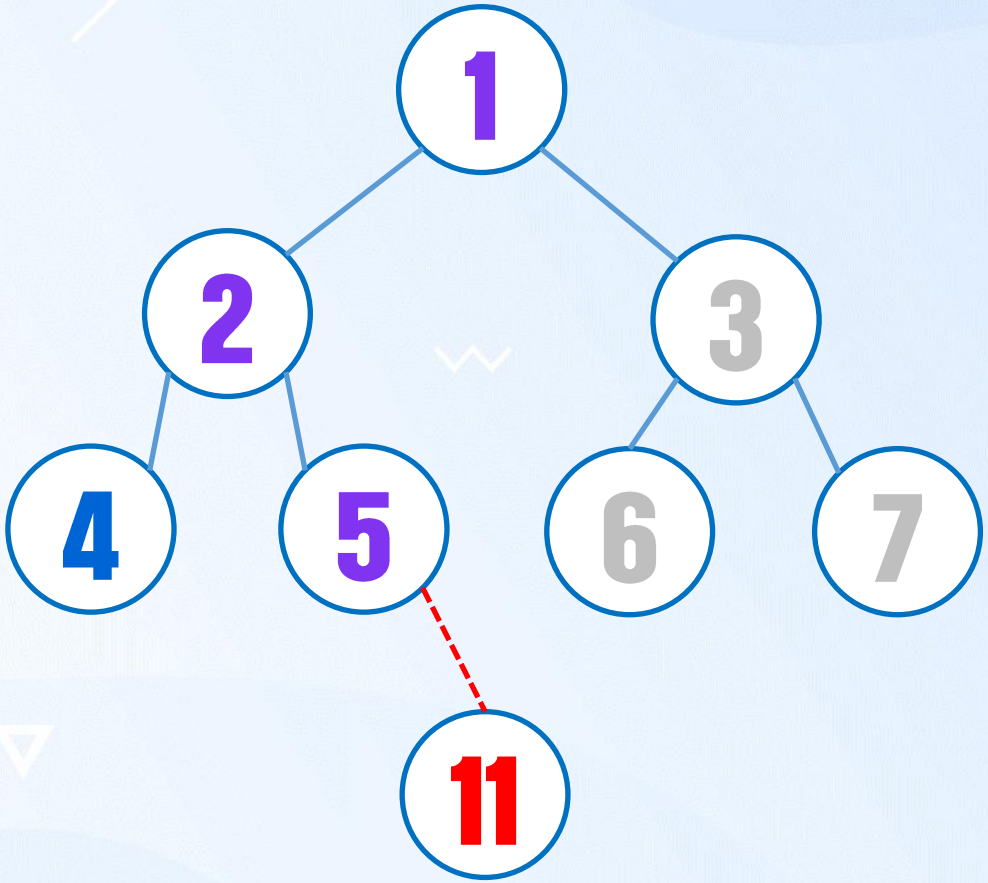
STACK

결과: 4

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=11



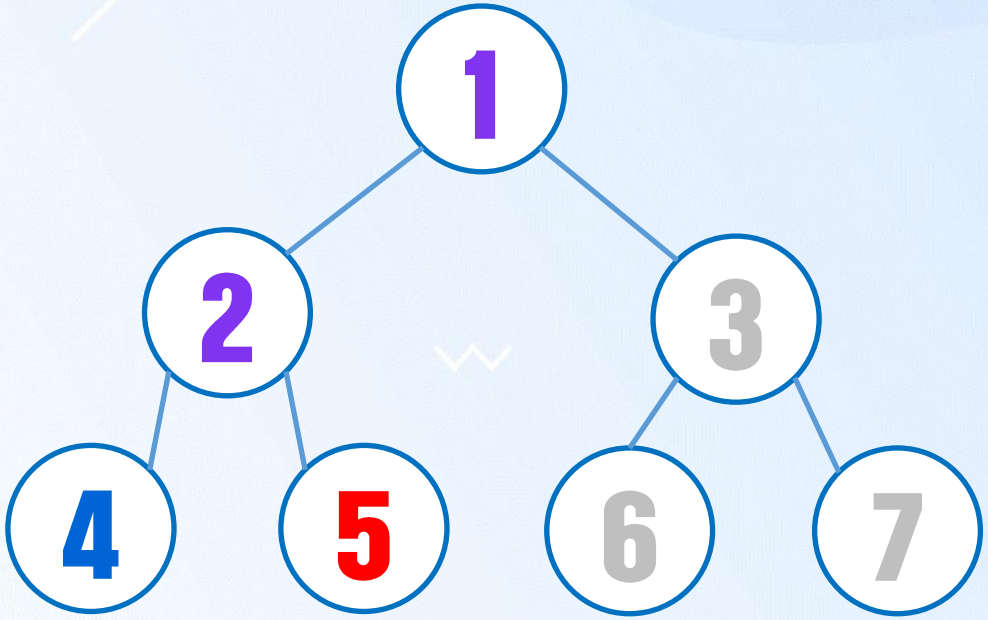


STACK

결과: 4

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {      n=5
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);   → 10
            DFS(n * 2 + 1); → 11
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```





결과: 4 → 5

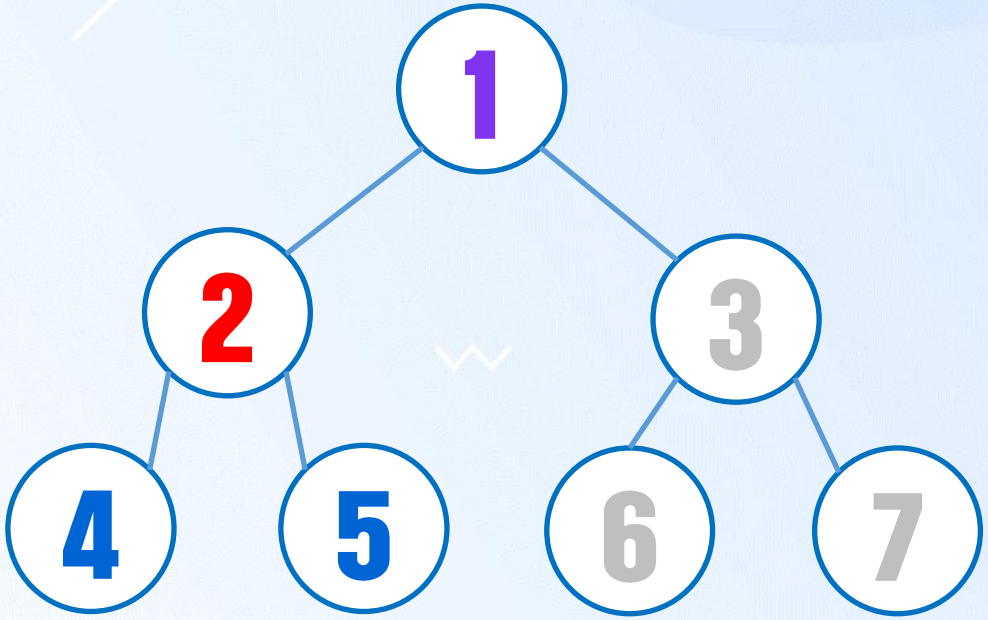
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=2

→ 4

→ 5





STACK

결과: 4 → 5 → 2

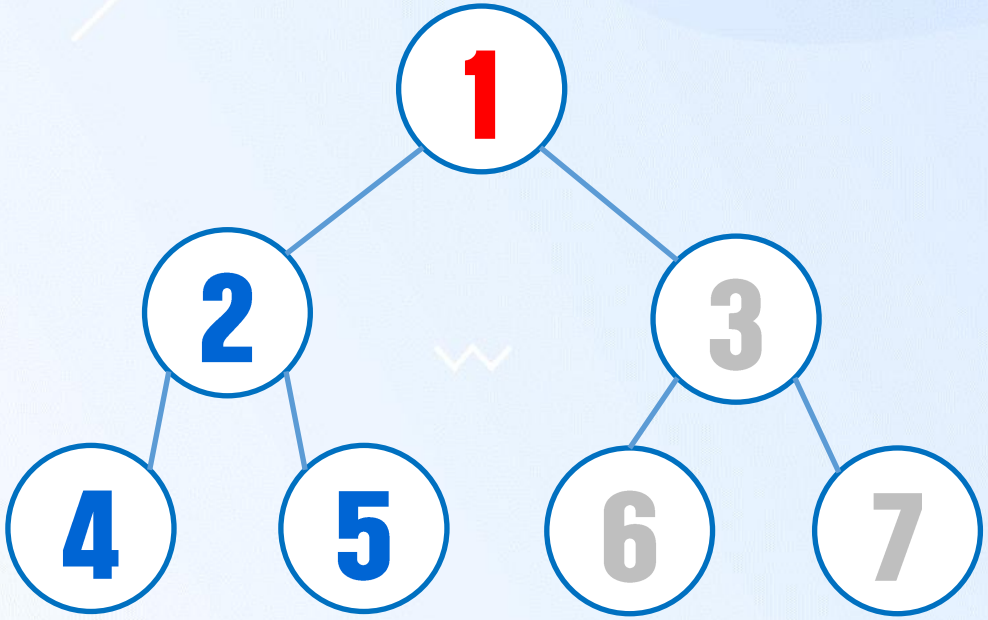
# 이진트리 후위순회

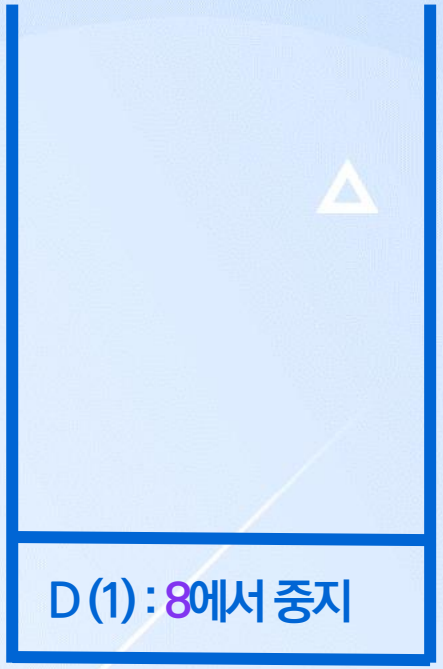
```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=1

→ 2

→ 3




STACK

결과: 4 → 5 → 2

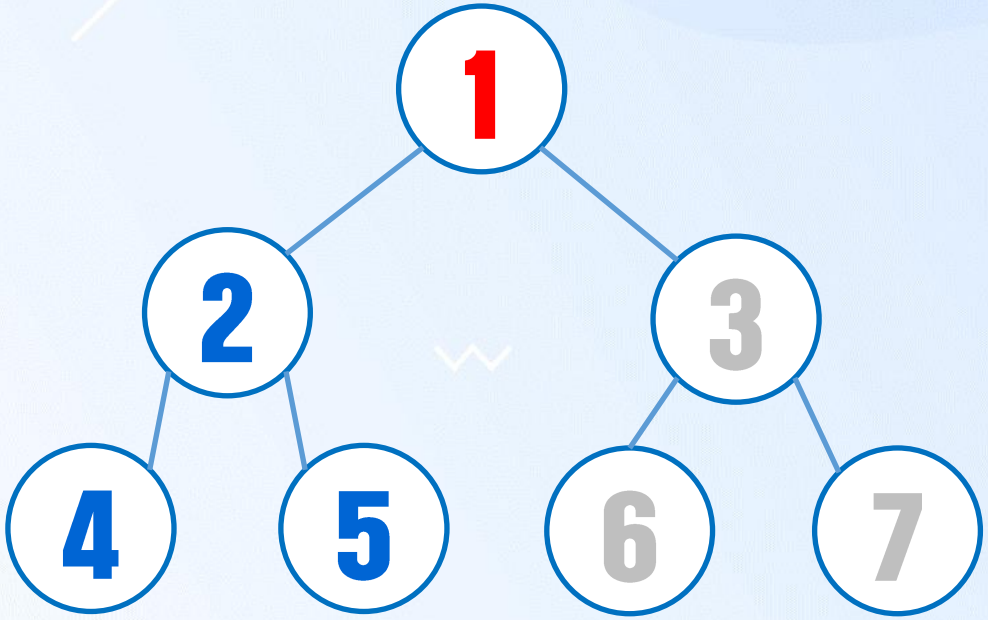
# 이진트리 후위순회

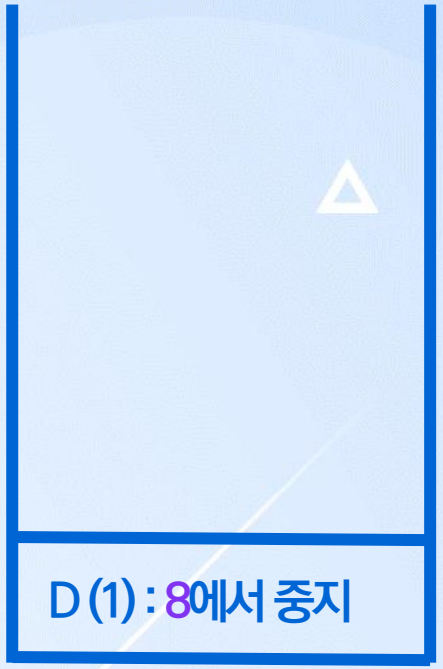
```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=1

→ 2

→ 3



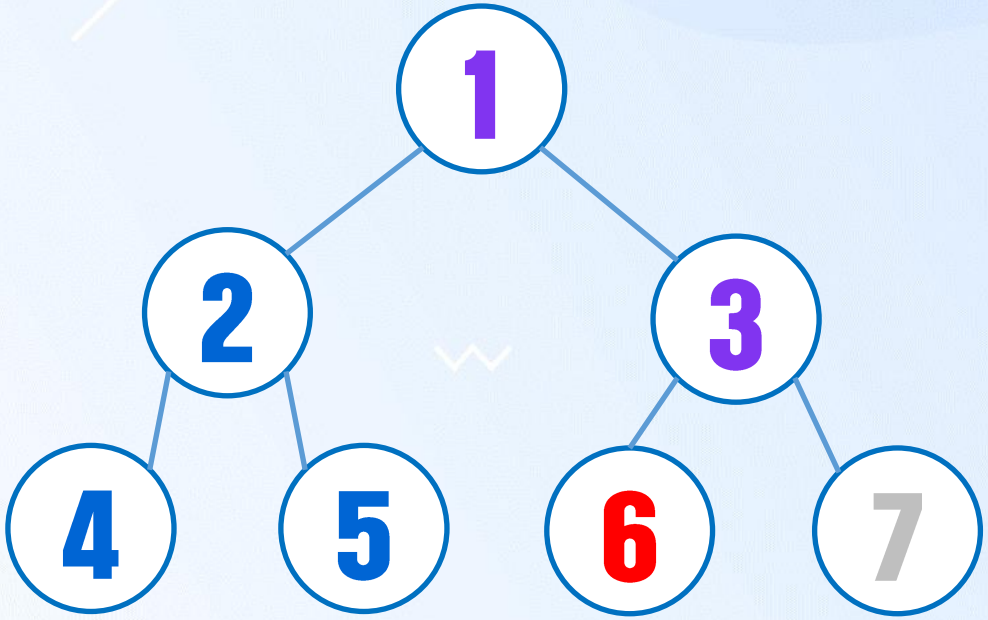

STACK

결과: 4 → 5 → 2

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);  → 12
            DFS(n * 2 + 1); → 13
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=6





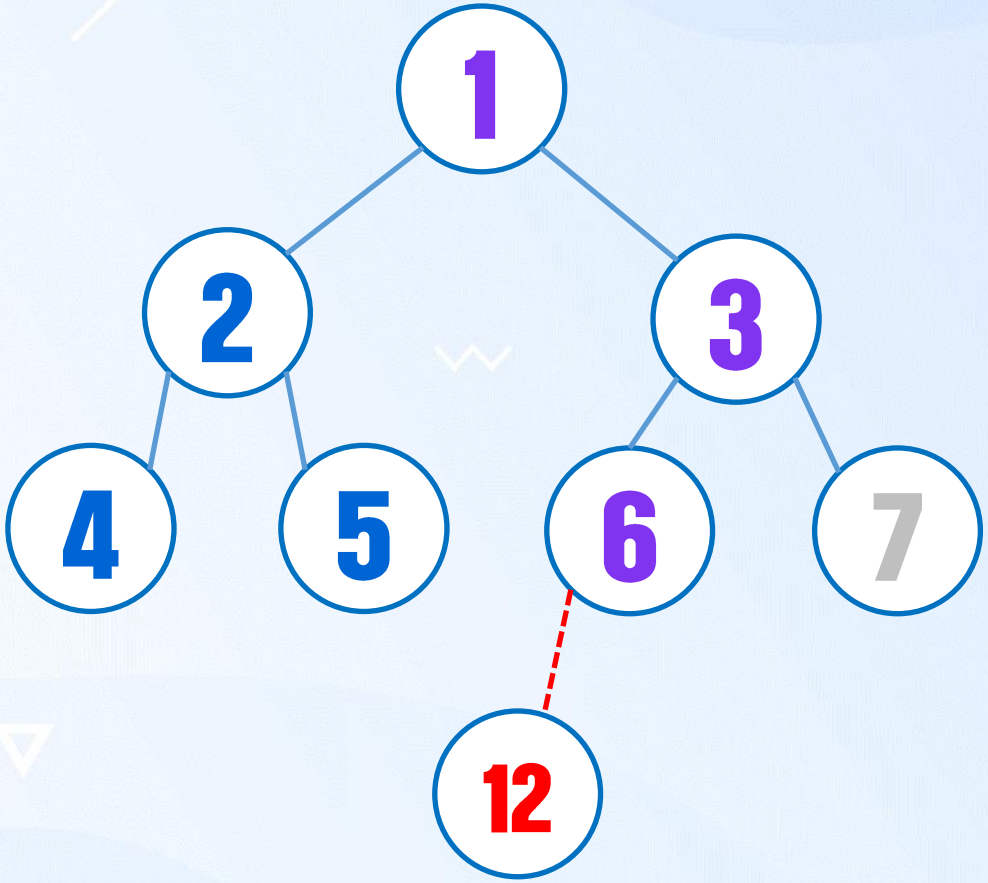
STACK

결과: 4 → 5 → 2

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=12





결과: 4 → 5 → 2

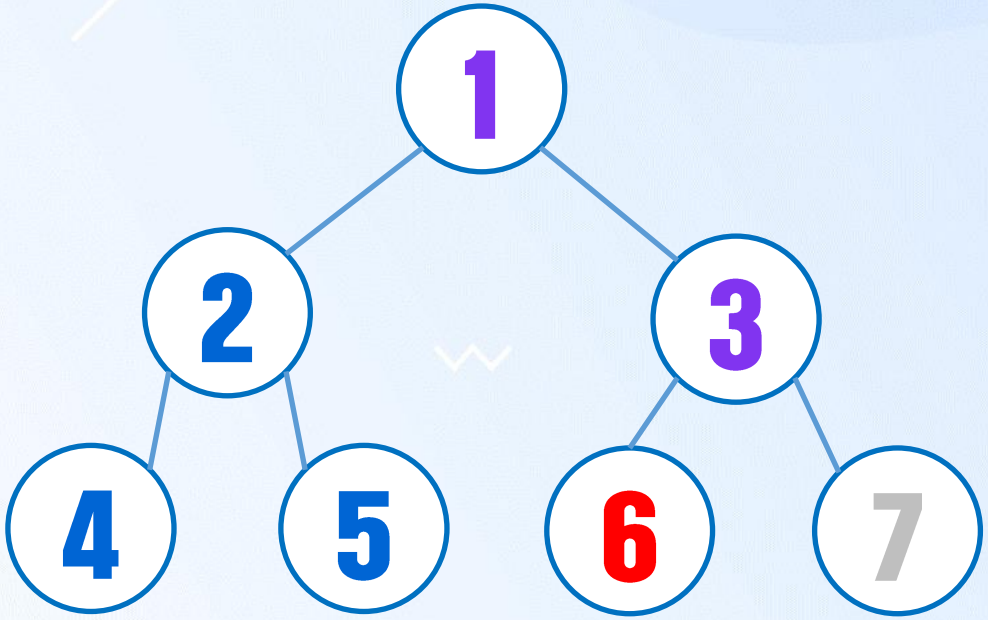
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=6

→ 12

→ 13





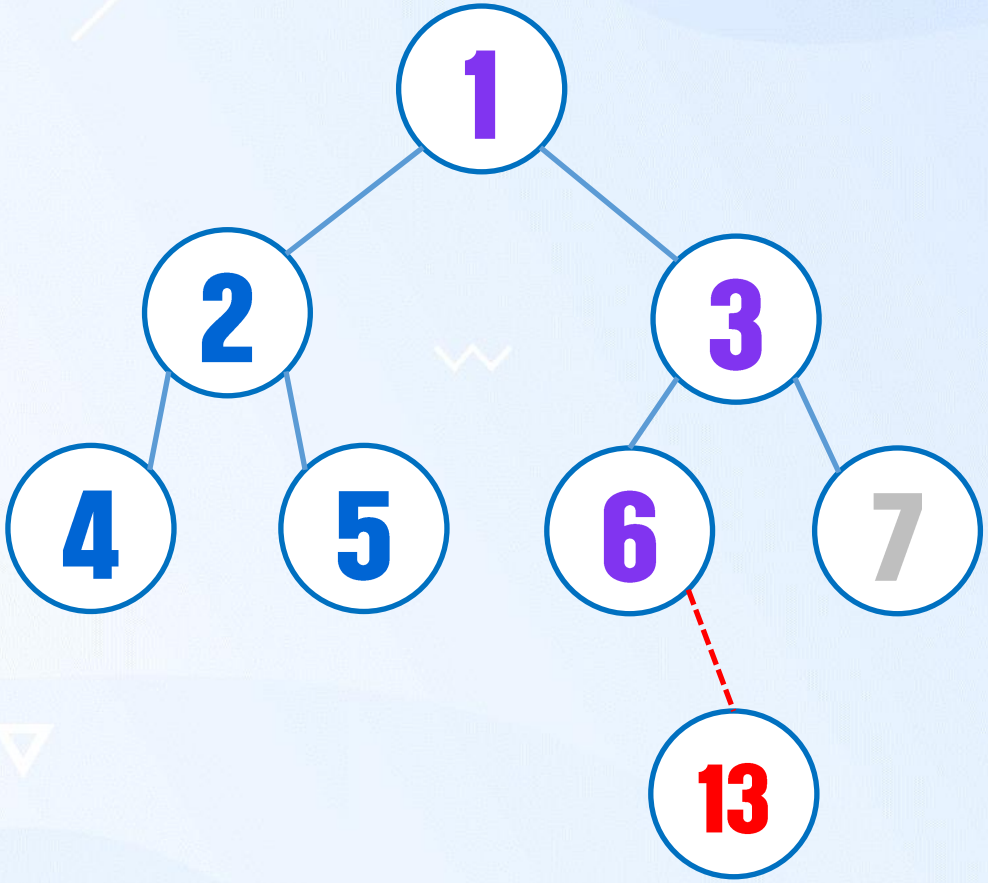
STACK

결과: 4 → 5 → 2

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=13

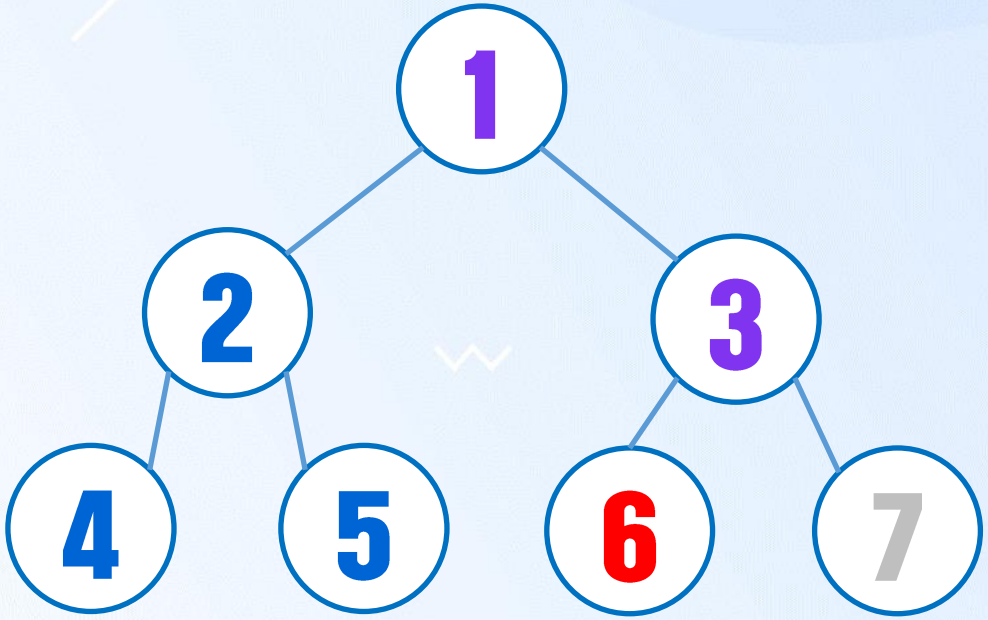




결과: 4 → 5 → 2

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {      n=6
        // 🏁 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);   → 12
            DFS(n * 2 + 1); → 13
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```



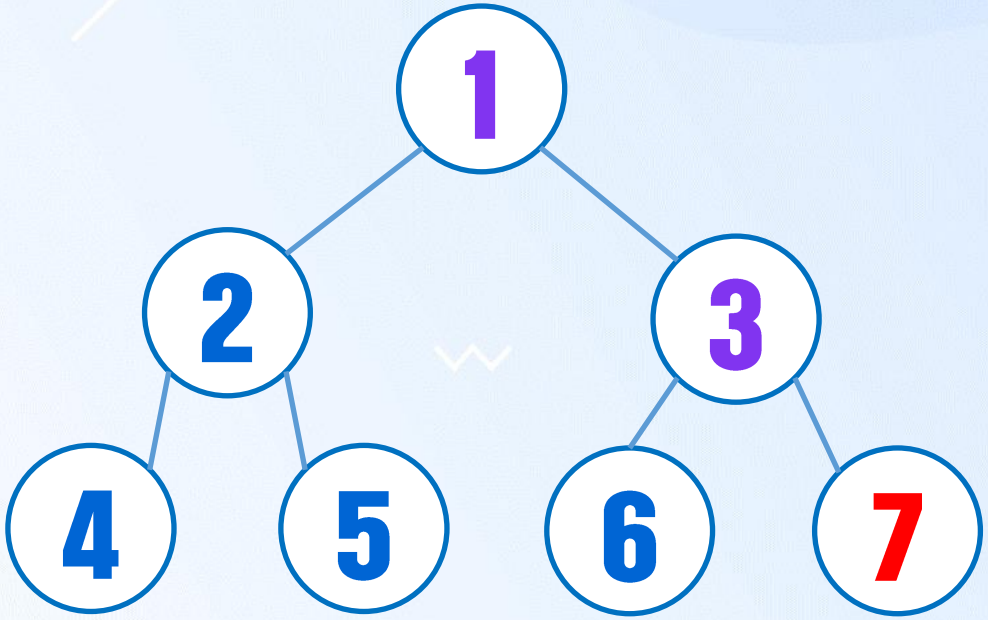


결과: 4 → 5 → 2 → 6

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);  → 14
            DFS(n * 2 + 1); → 15
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=7





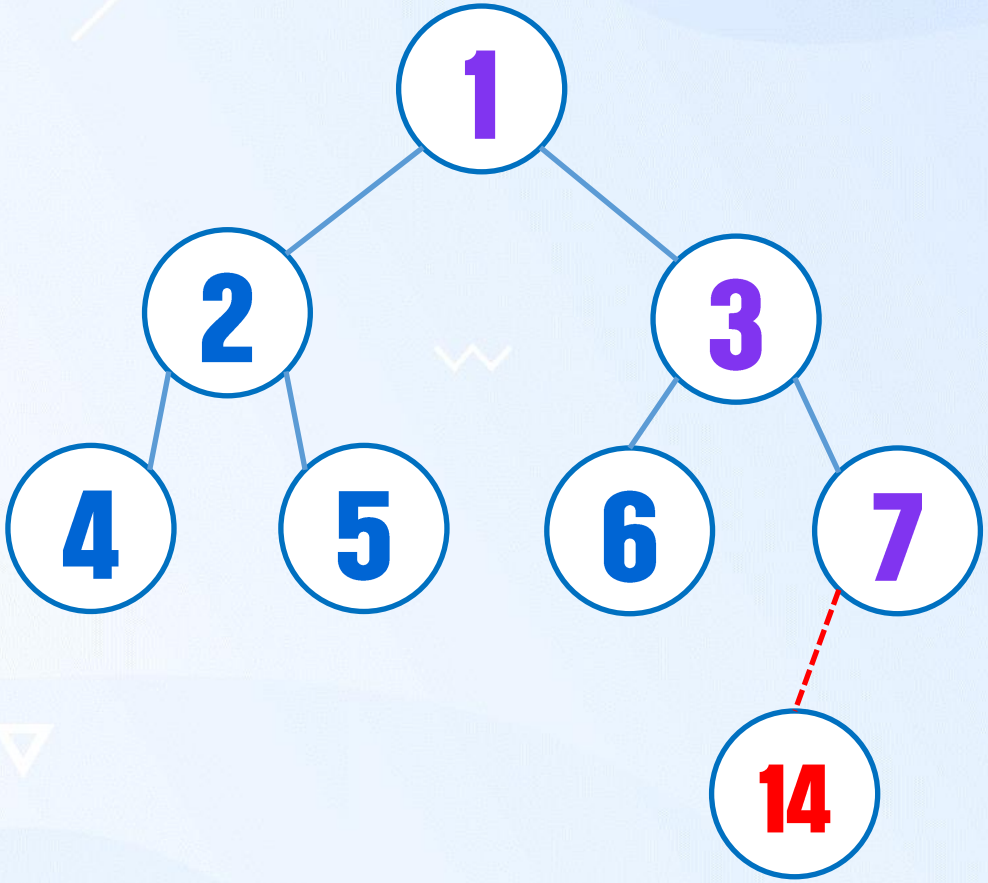
STACK

결과: 4 → 5 → 2 → 6

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=14





결과: 4 → 5 → 2 → 6

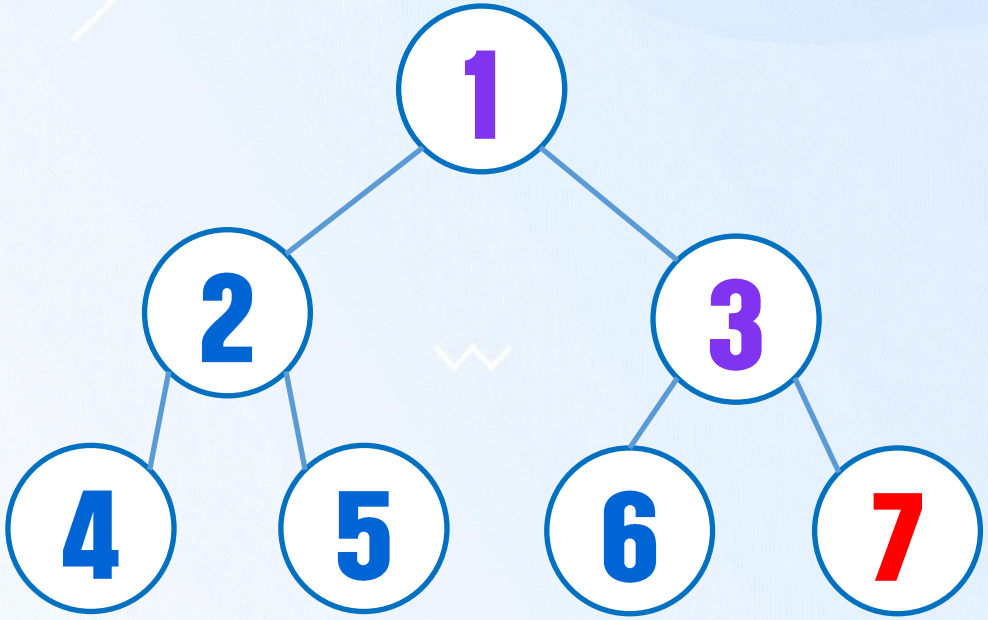
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=7

DFS(n * 2); → 14

DFS(n * 2 + 1); → 15





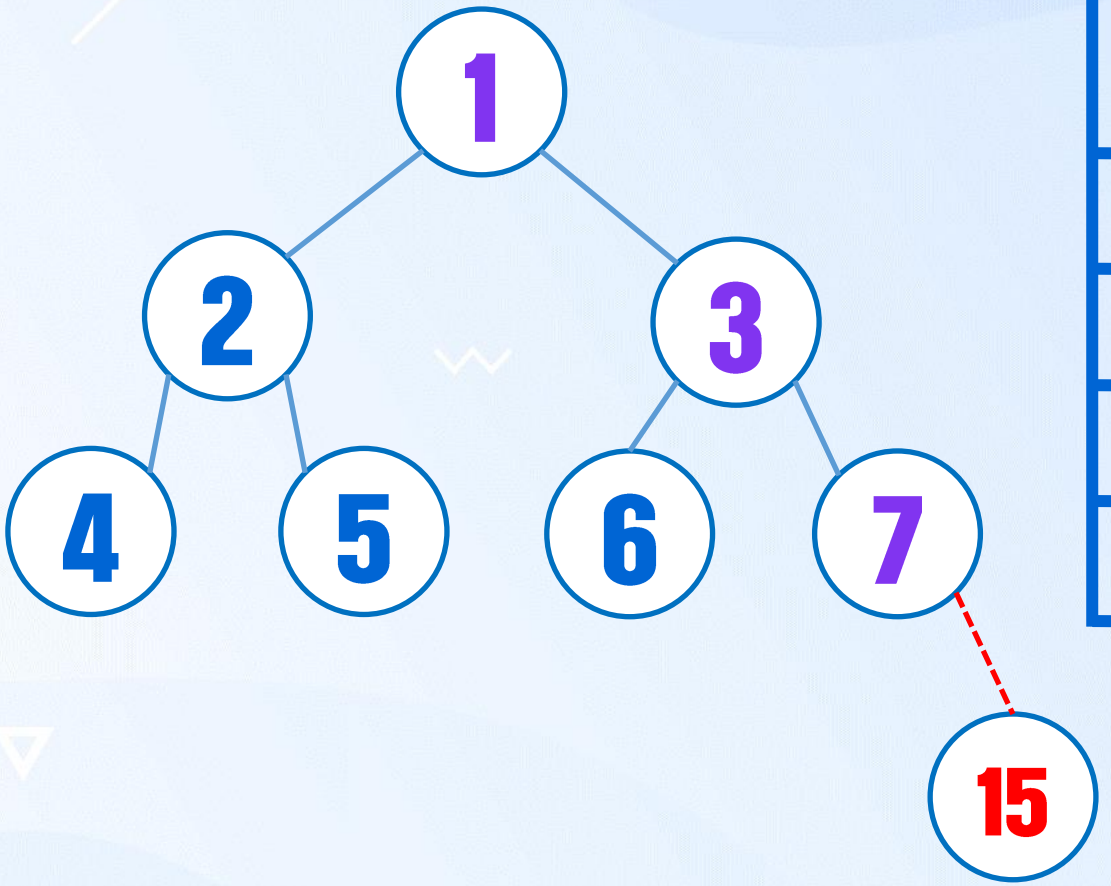
STACK

결과: 4 → 5 → 2 → 6

# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=15





결과: 4 → 5 → 2 → 6

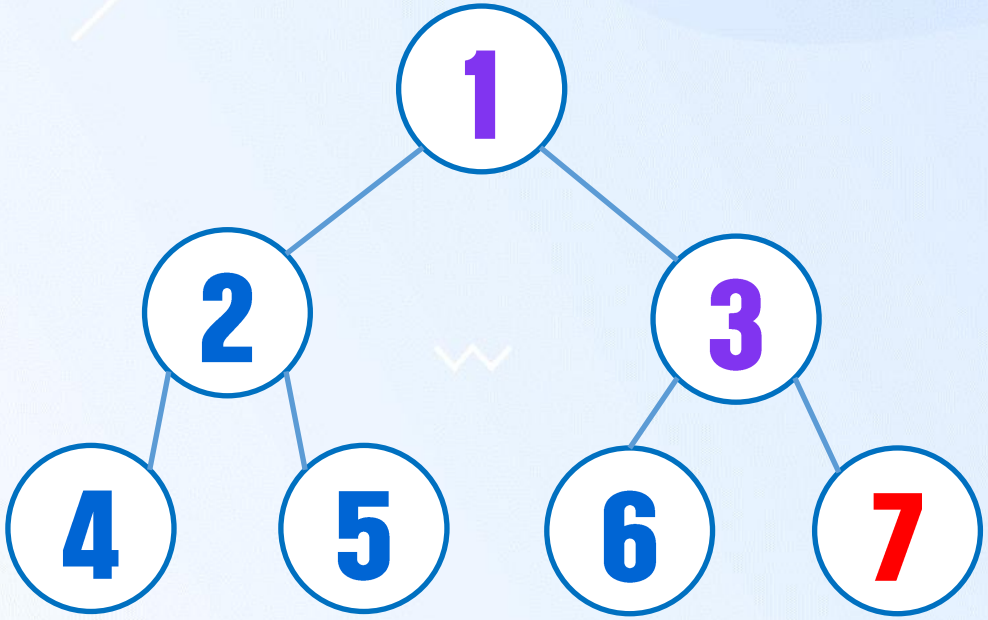
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=7

DFS(n * 2); → 14

DFS(n * 2 + 1); → 15





STACK

결과: 4 → 5 → 2 → 6 → 7

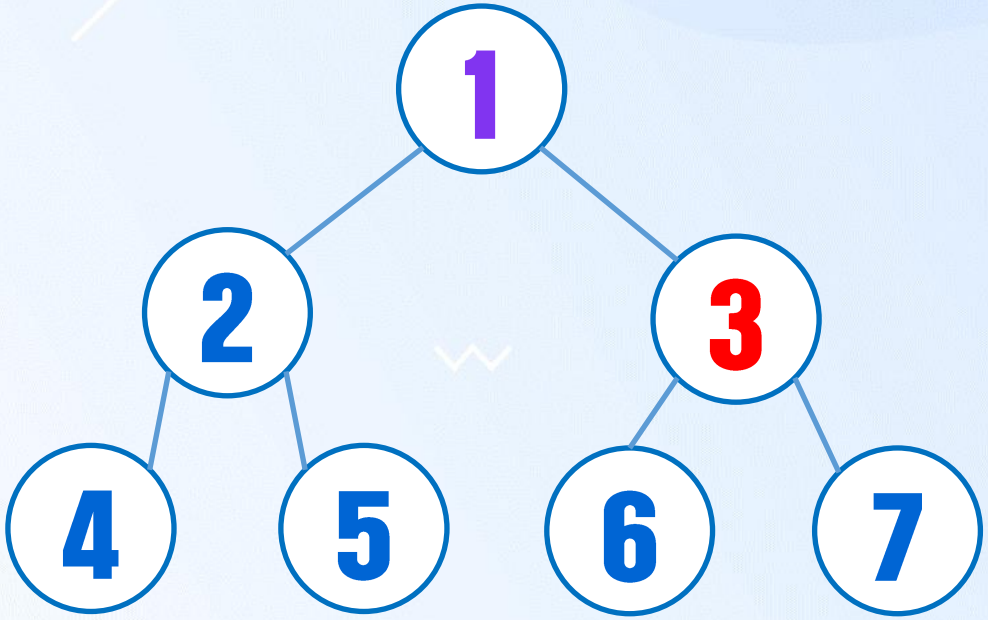
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=3

→ 6

→ 7





STACK

결과: 4 → 5 → 2 → 6 → 7 → 3

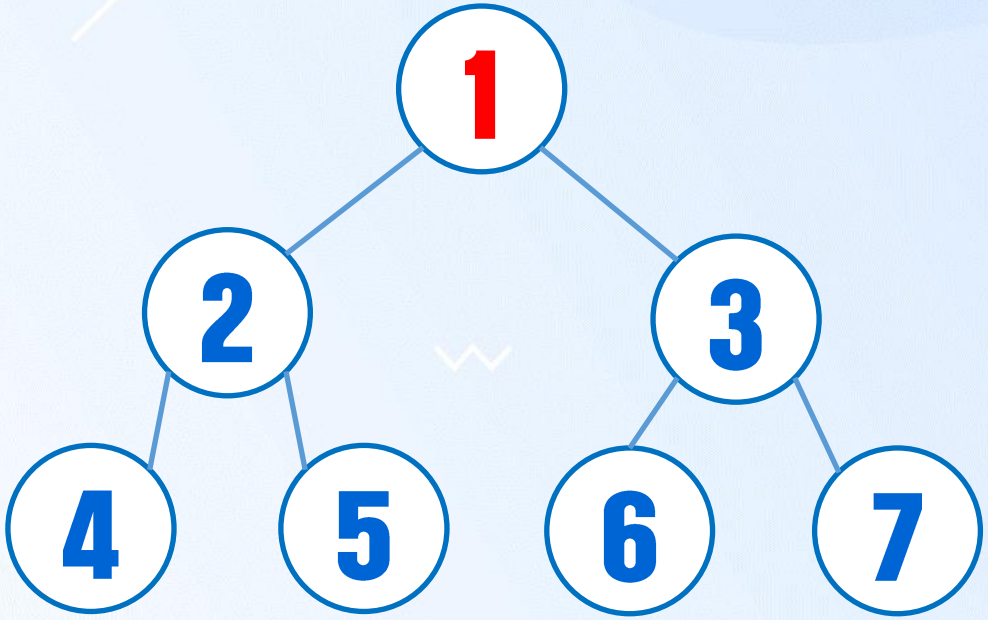
# 이진트리 후위순회

```
function solution(n) {
    function DFS(n) {
        // 종착 지점
        if (n > 7) return;
        else {
            // 계속 트리가 뻗어나가는 지점
            DFS(n * 2);
            DFS(n * 2 + 1);
            console.log(n); // 후위 순회
        }
    }
    DFS(n);
}
solution(1);
```

n=1

→ 2

→ 3





결과: 4 → 5 → 2 → 6 → 7 → 3 → 1

# 감사합니다