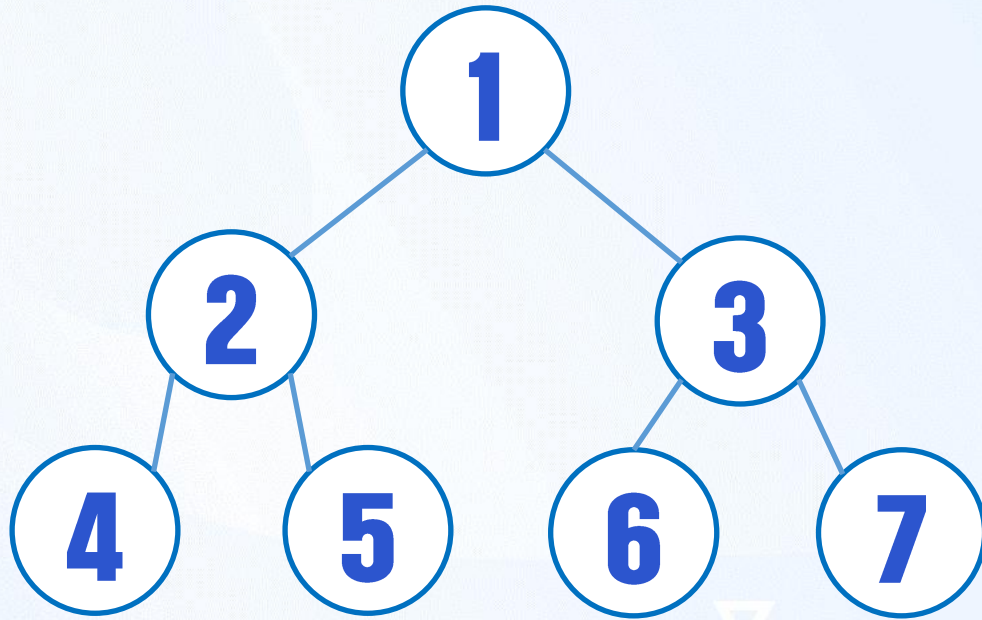


이진 트리 후위순회



이진트리 후위순회

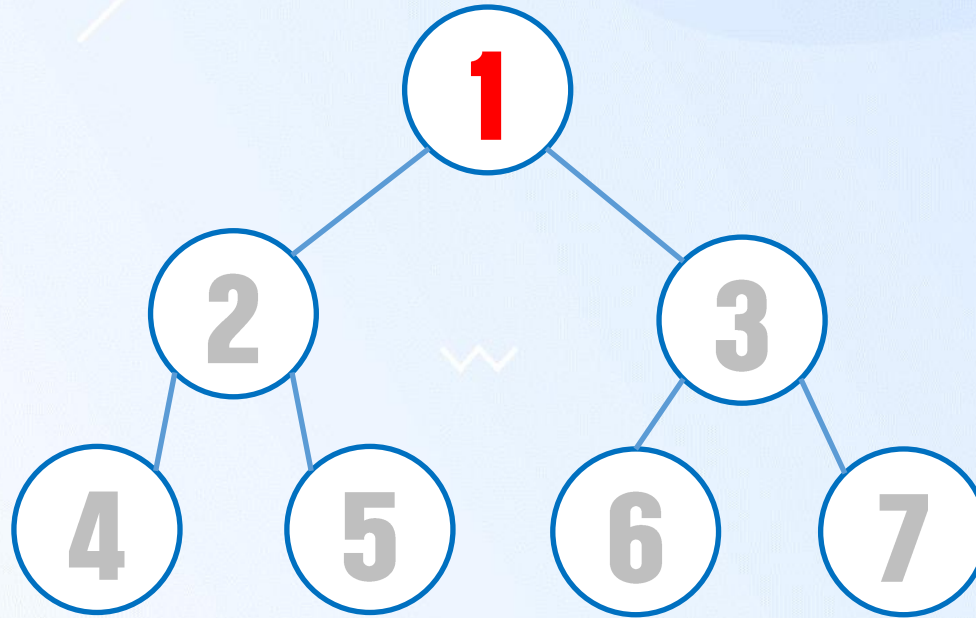


- 자식(좌) → 자식(우) → 부모 노드 순으로 순회
- 재귀 호출을 이용
- 부모가 n 이라면
 - 자식(좌) = $n * 2$
 - 자식(우) = $n * 2 + 1$

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=1  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뻗어나가는 지점  
7       ✓ DFS(n * 2); → 2  
8         DFS(n * 2 + 1); → 3  
9         console.log(n); // 후위 순회  
10      }  
11    }  
12    DFS(n);  
13  }  
14  solution(1);
```



D(1): 7에서 중지

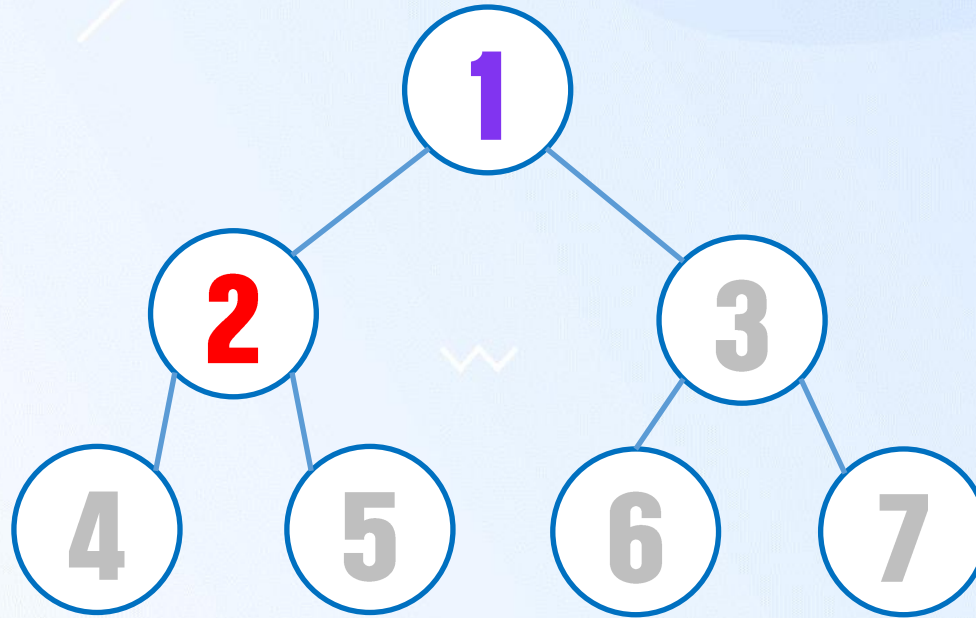
STACK

결과:

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=2  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       ✓ DFS(n * 2); → 4  
8         DFS(n * 2 + 1); → 5  
9         console.log(n); // 후위 순회  
10      }  
11    }  
12    DFS(n);  
13  }  
14  solution(1);
```



D (2): 7에서 중지

D (1): 7에서 중지

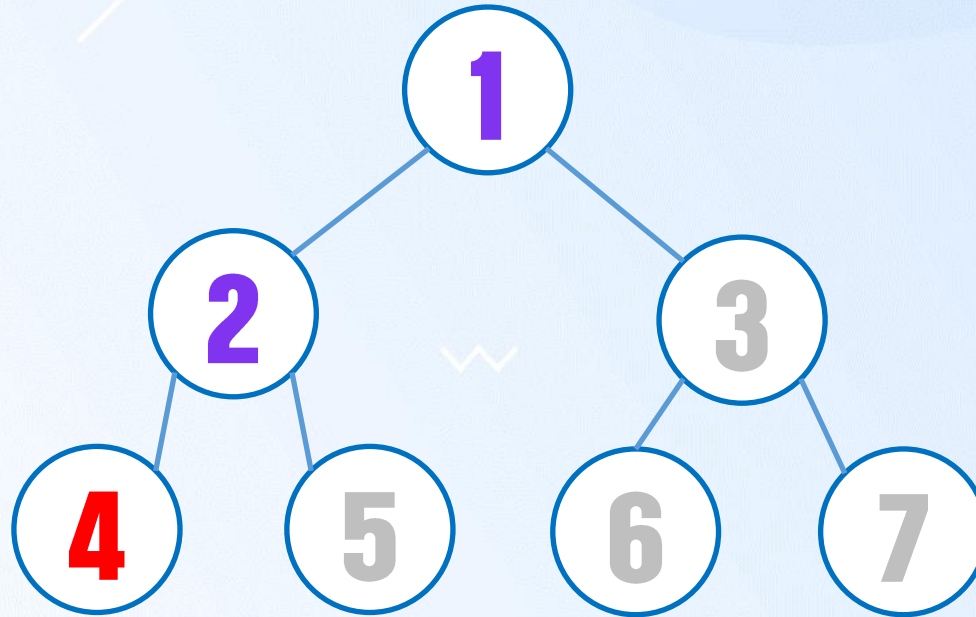
STACK

결과:

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=4  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뻗어나가는 지점  
7       ✓ DFS(n * 2); → 8  
8         DFS(n * 2 + 1); → 9  
9         console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (4): 7에서 중지

D (2): 7에서 중지

D (1): 7에서 중지

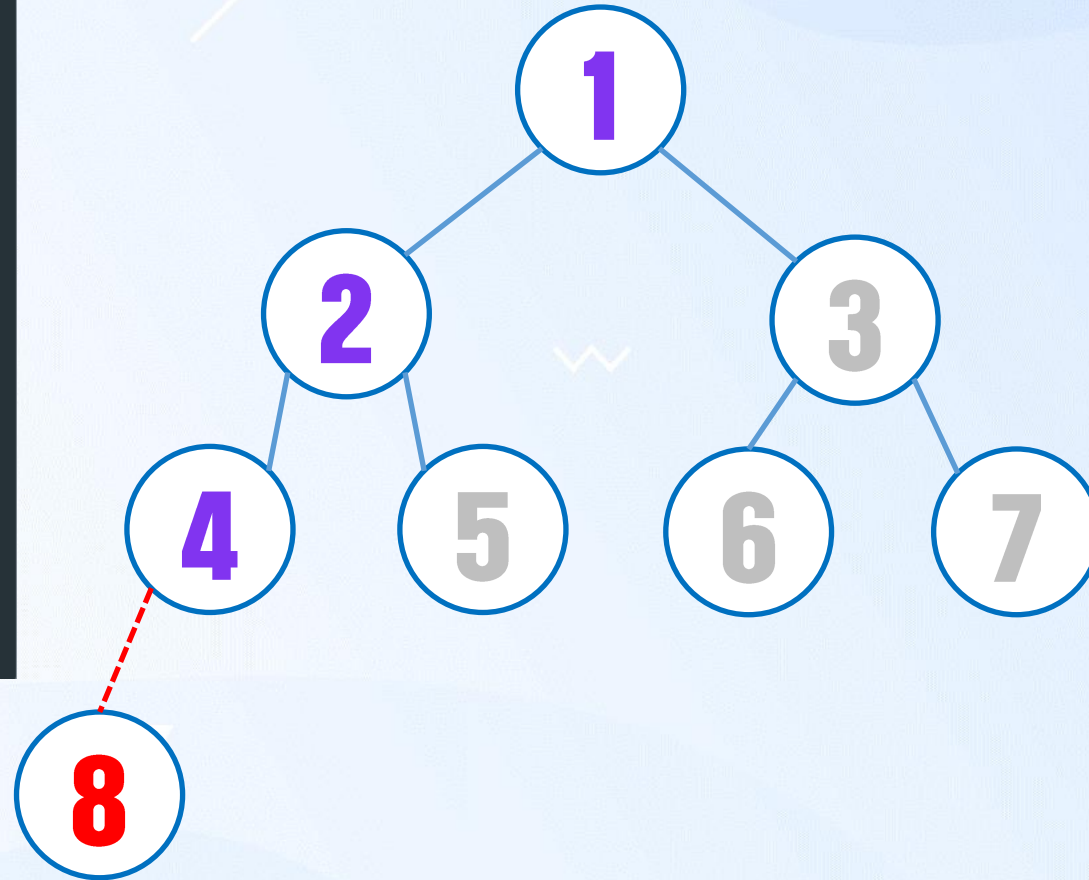
STACK

결과:

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=8  
3     // 🚧 종착 지점  
4     ✓ if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2);  
8       DFS(n * 2 + 1);  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (8) : pop

D (4) : 7에서 중지

D (2) : 7에서 중지

D (1) : 7에서 중지

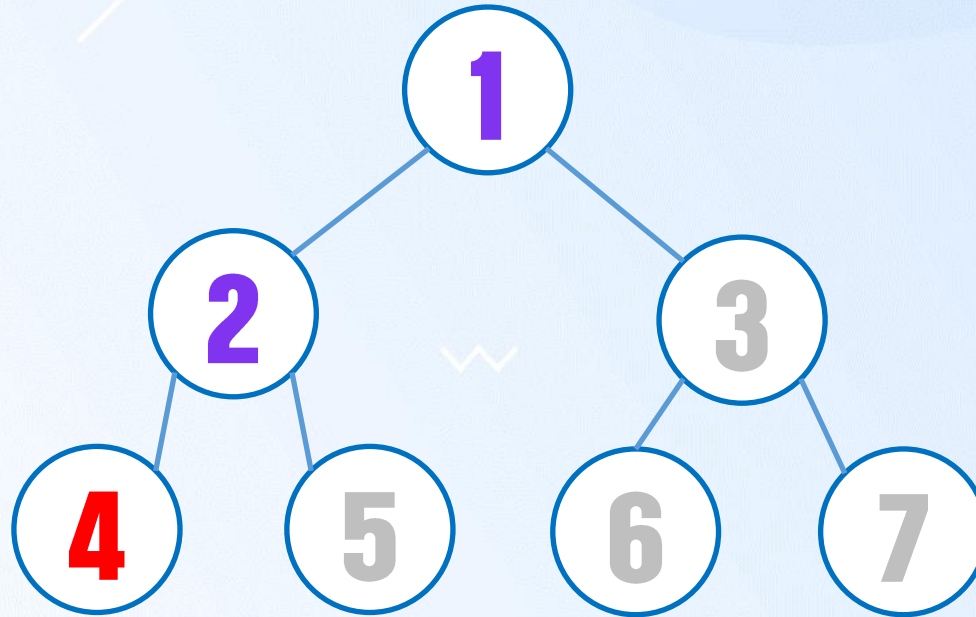
STACK

결과:

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=4  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 8  
8       ✓ DFS(n * 2 + 1); → 9  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (4): 8에서 중지

D (2): 7에서 중지

D (1): 7에서 중지

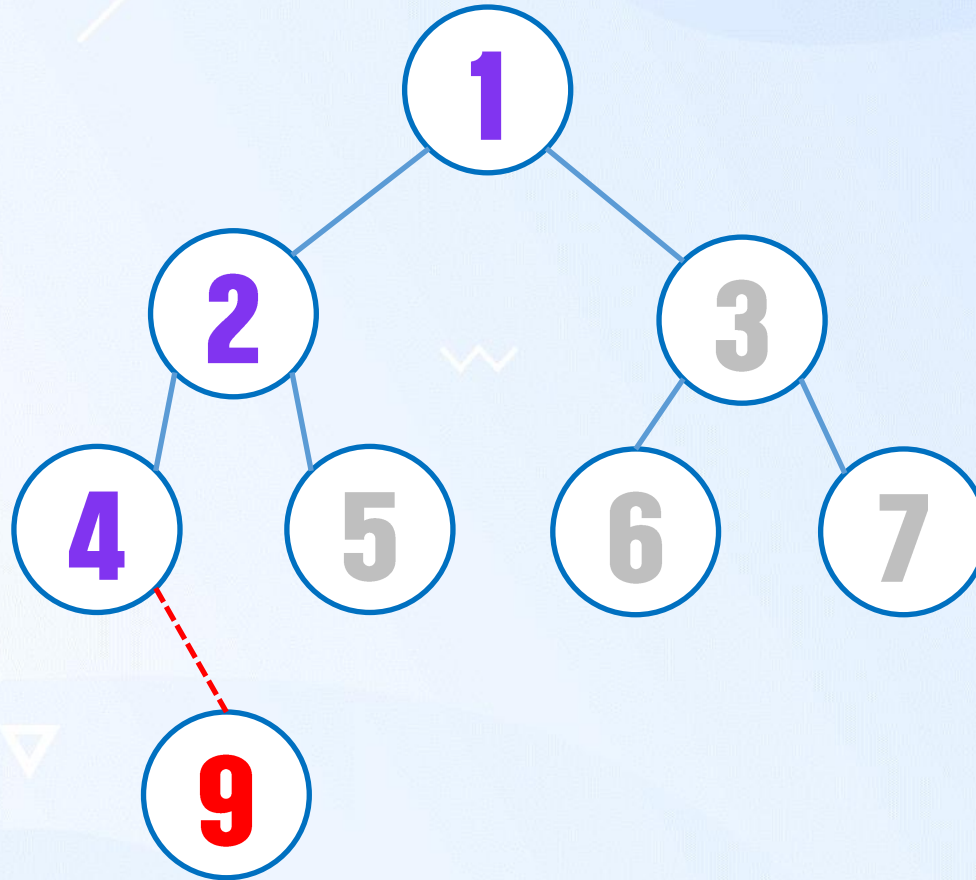
STACK

결과:

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=9  
3     // 🚧 종착 지점  
4     ✓ if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2);  
8       DFS(n * 2 + 1);  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (9) : pop

D (4) : 8에서 중지

D (2) : 7에서 중지

D (1) : 7에서 중지

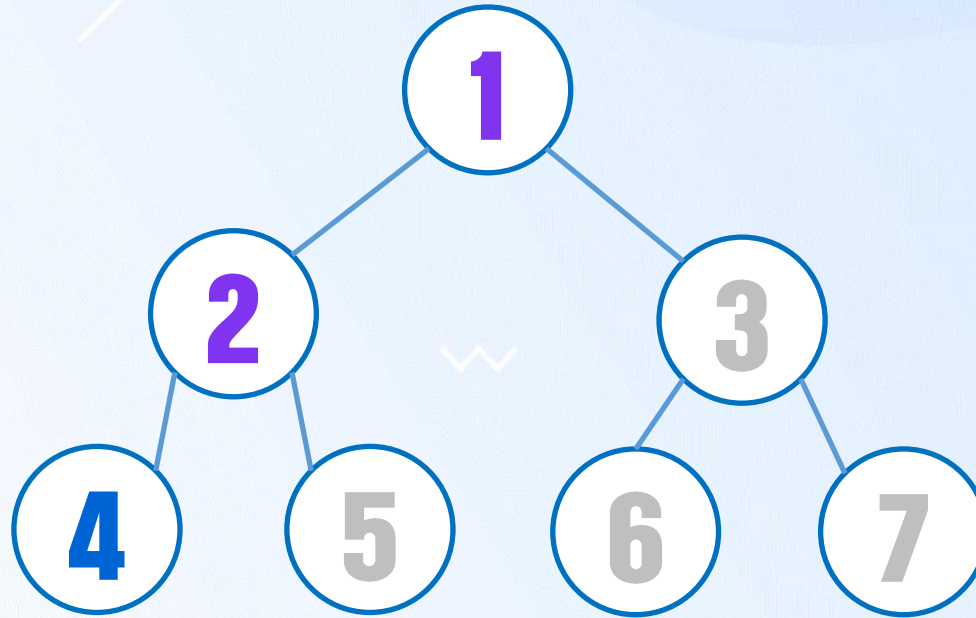
STACK

결과:

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=4  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 8  
8       DFS(n * 2 + 1); → 9  
9       ✓ console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (4): pop

D (2): 7에서 중지

D (1): 7에서 중지

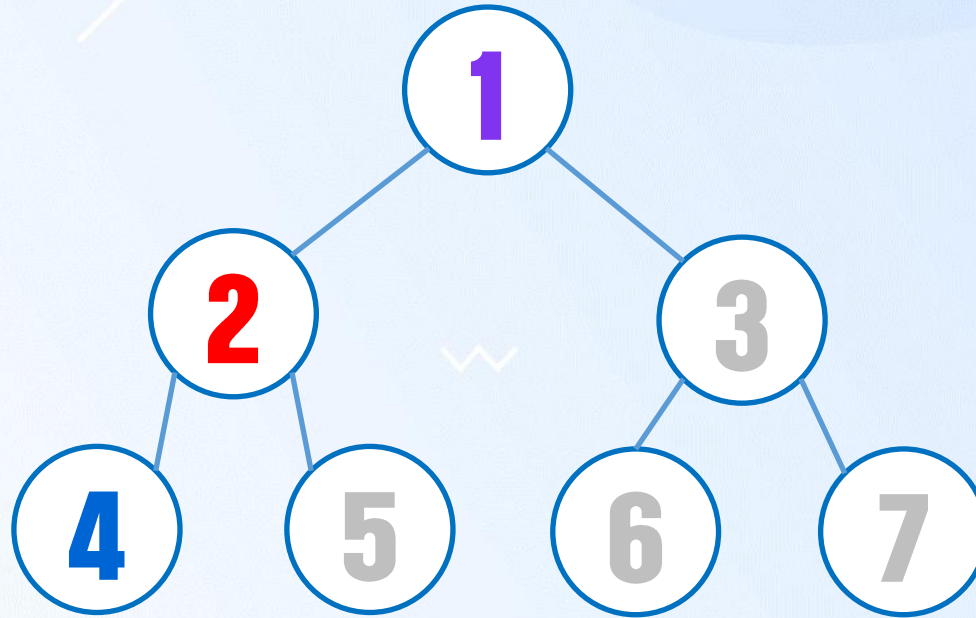
STACK

결과: 4

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=2  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 4  
8       ✓ DFS(n * 2 + 1); → 5  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (2): 8에서 중지

D (1): 7에서 중지

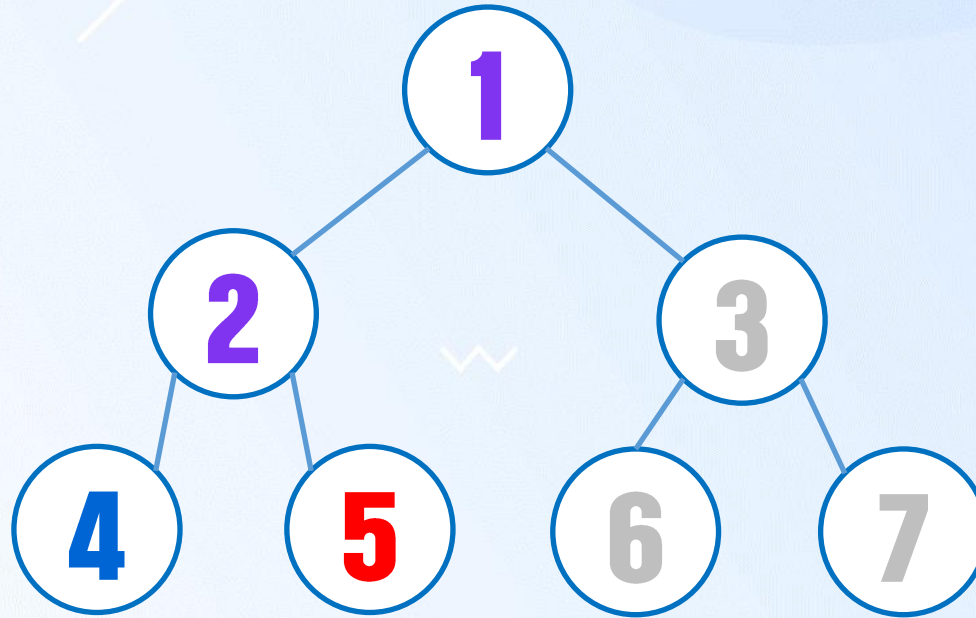
STACK

결과: 4

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=5  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       ✓ DFS(n * 2); → 10  
8         DFS(n * 2 + 1); → 11  
9         console.log(n); // 후위 순회  
10      }  
11    }  
12    DFS(n);  
13  }  
14  solution(1);
```



D (5): 7에서 중지

D (2): 8에서 중지

D (1): 7에서 중지

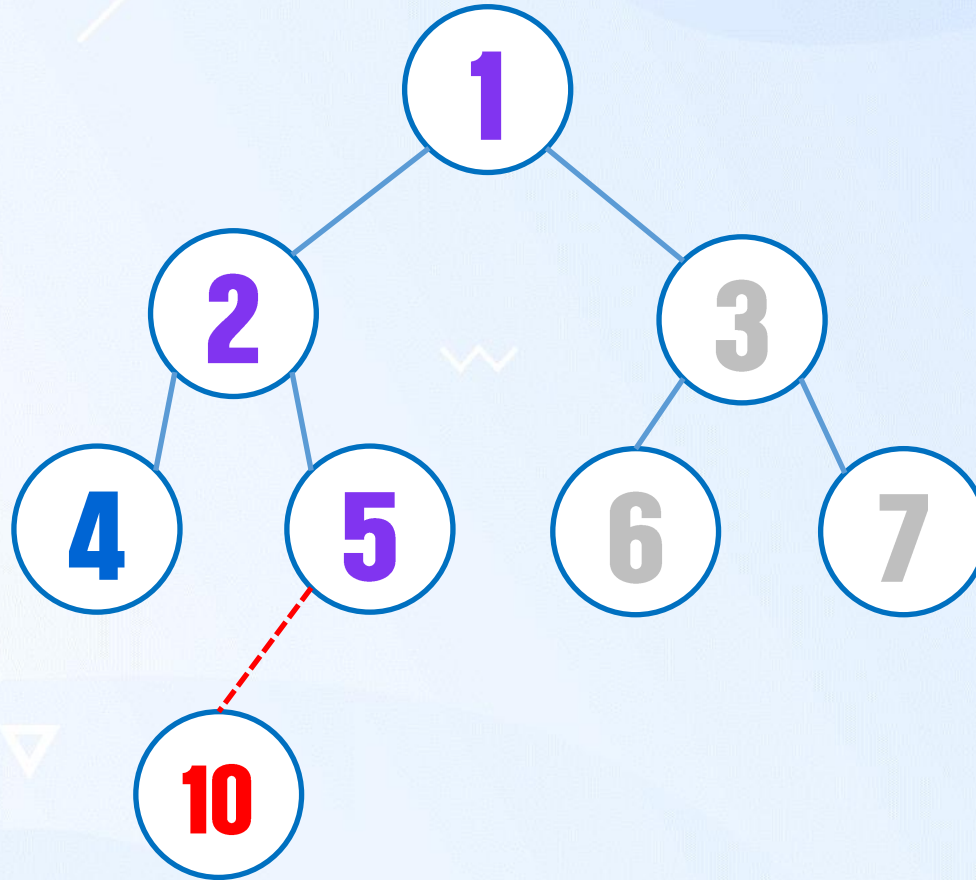
STACK

결과: 4

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=10  
3     // 🚧 종착 지점  
4     ✓ if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2);  
8       DFS(n * 2 + 1);  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (10) : pop

D (5) : 7에서 중지

D (2) : 7에서 중지

D (1) : 7에서 중지

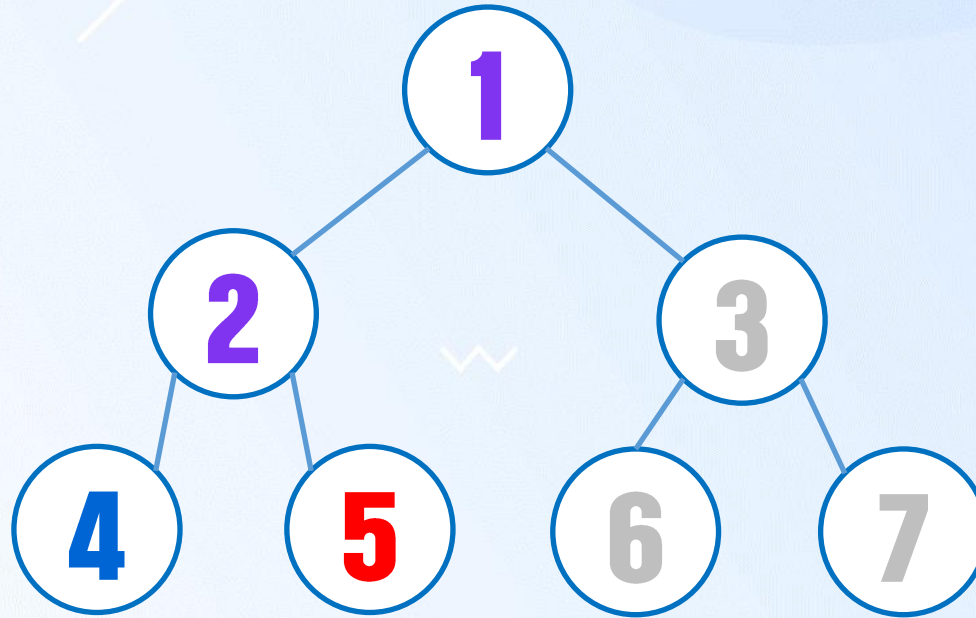
STACK

결과: 4

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=5  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 10  
8       ✓ DFS(n * 2 + 1); → 11  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (5): 8에서 중지

D (2): 8에서 중지

D (1): 7에서 중지

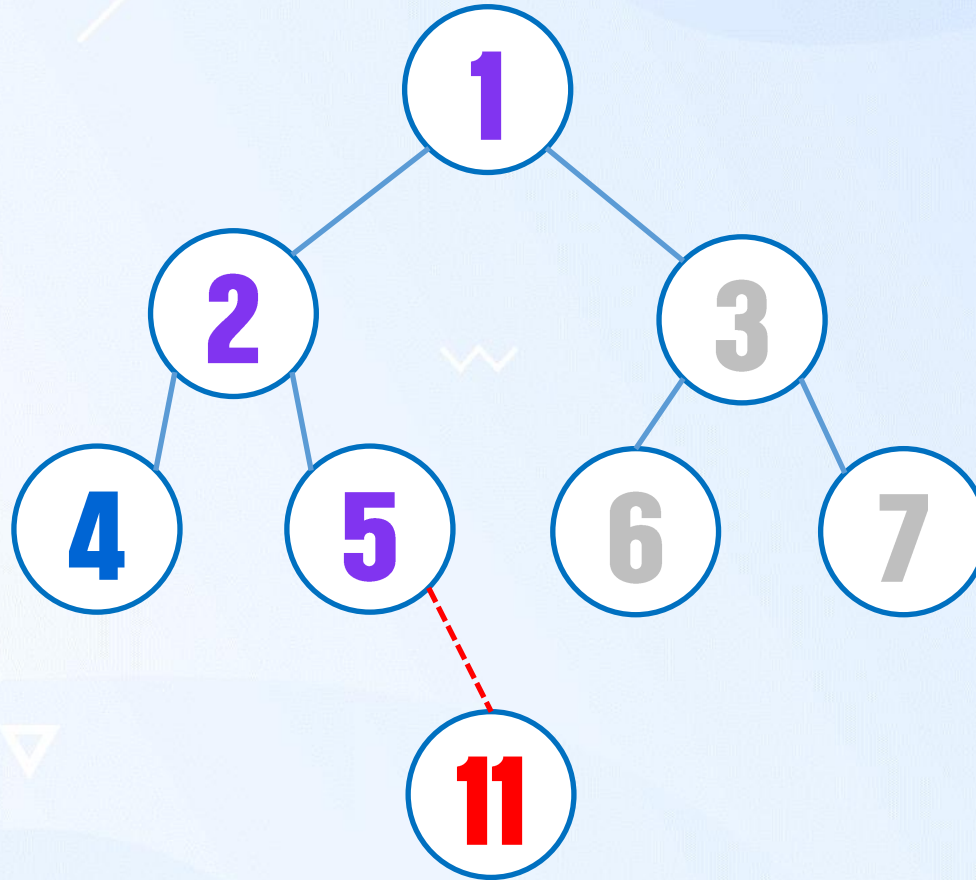
STACK

결과: 4

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=11  
3     // 🚧 종착 지점  
4     ✓ if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2);  
8       DFS(n * 2 + 1);  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (11): pop

D (5): 8에서 중지

D (2): 7에서 중지

D (1): 7에서 중지

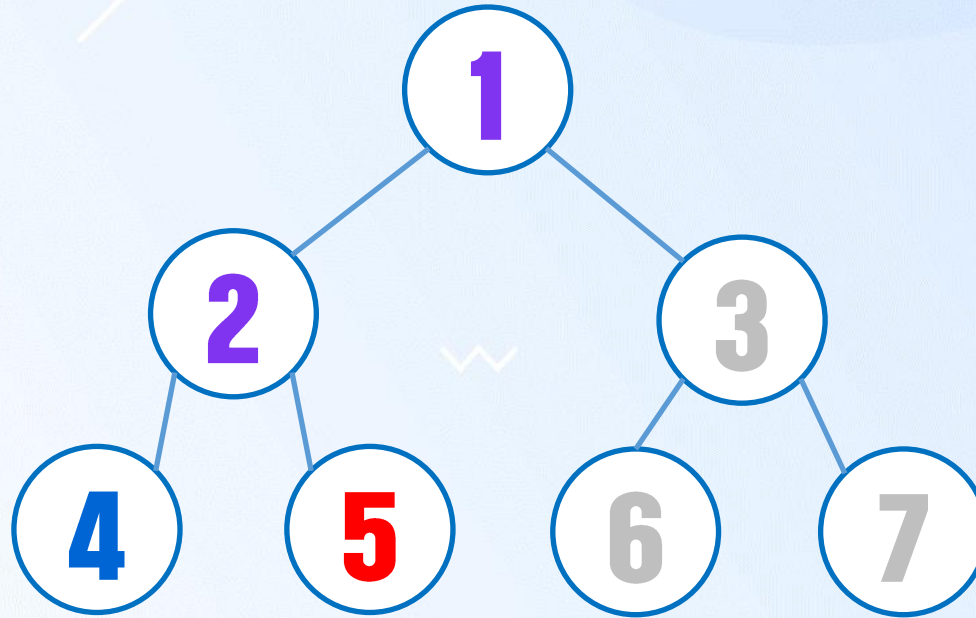
STACK

결과: 4

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=5  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 10  
8       DFS(n * 2 + 1); → 11  
9       ✓ console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (5) : pop

D (2) : 8에서 중지

D (1) : 7에서 중지

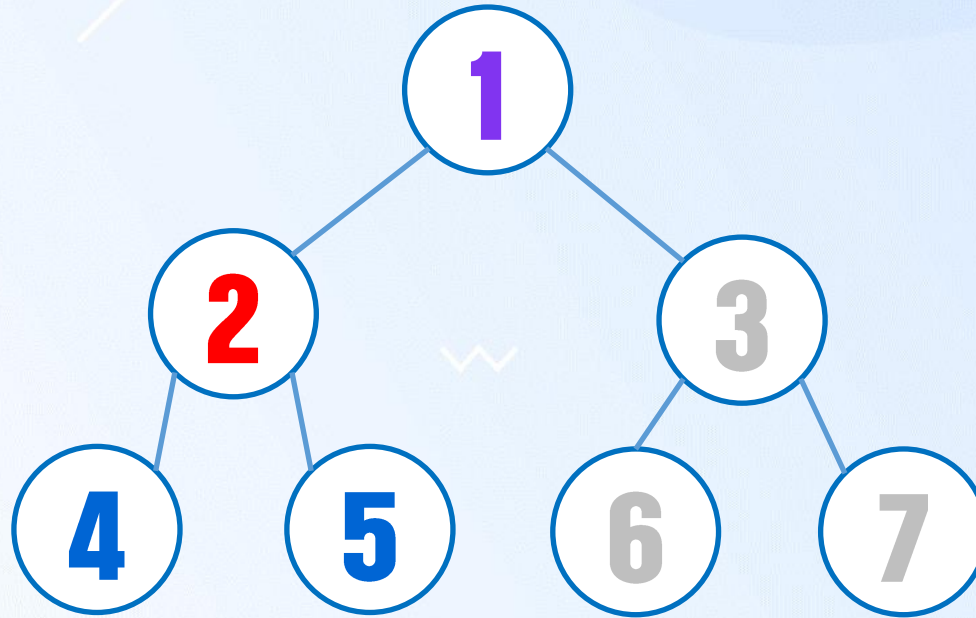
STACK

결과: 4 → 5

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=2  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 4  
8       DFS(n * 2 + 1); → 5  
9       ✓ console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



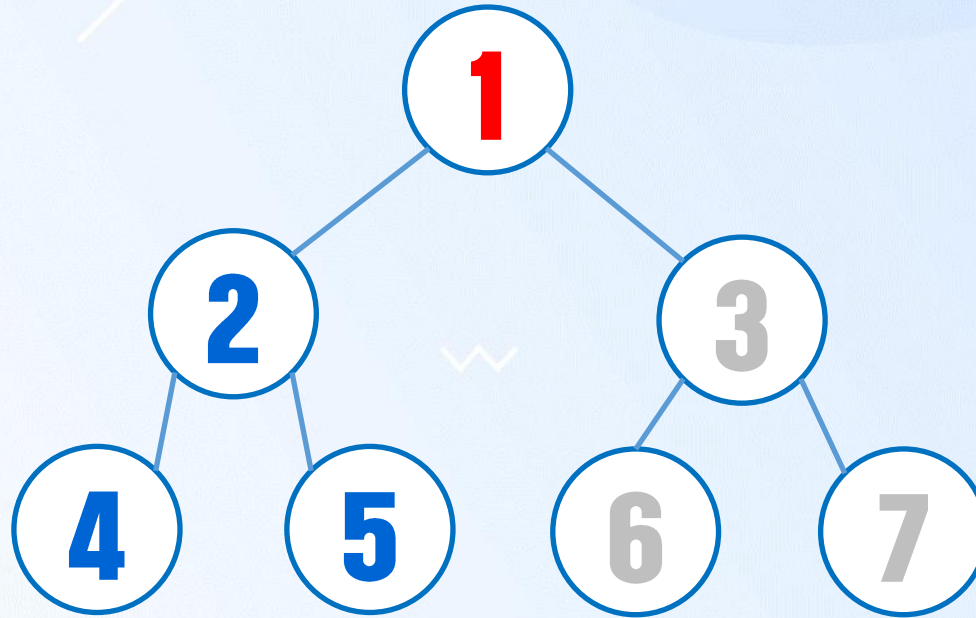
STACK

결과: 4 → 5 → 2

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=1  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뻗어나가는 지점  
7       DFS(n * 2); → 2  
8       ✓ DFS(n * 2 + 1); → 3  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D(1): 8에서 중지

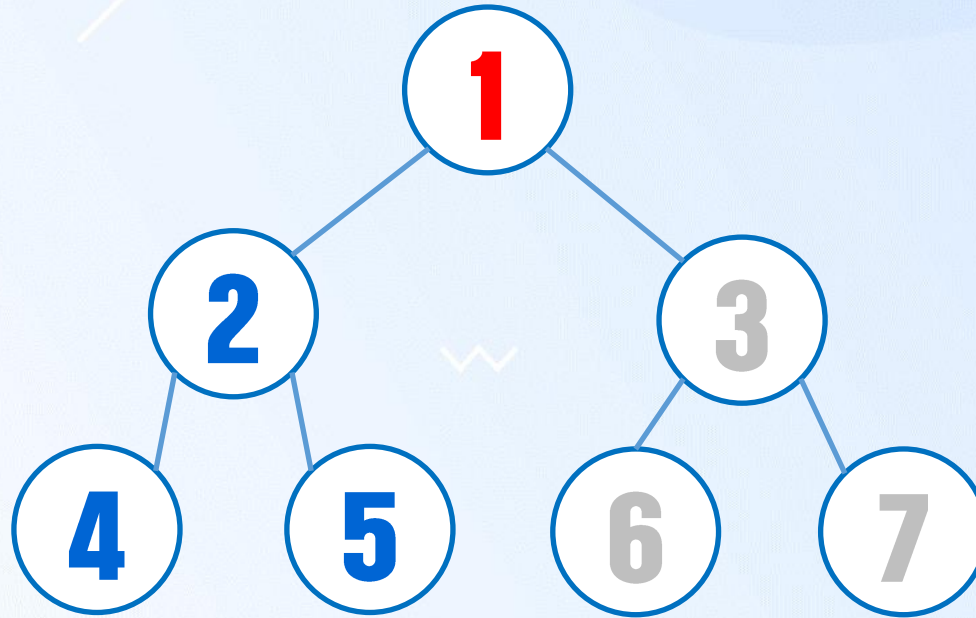
STACK

결과: 4 → 5 → 2

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=1  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뻗어나가는 지점  
7       DFS(n * 2); → 2  
8       ✓ DFS(n * 2 + 1); → 3  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D(1): 8에서 중지

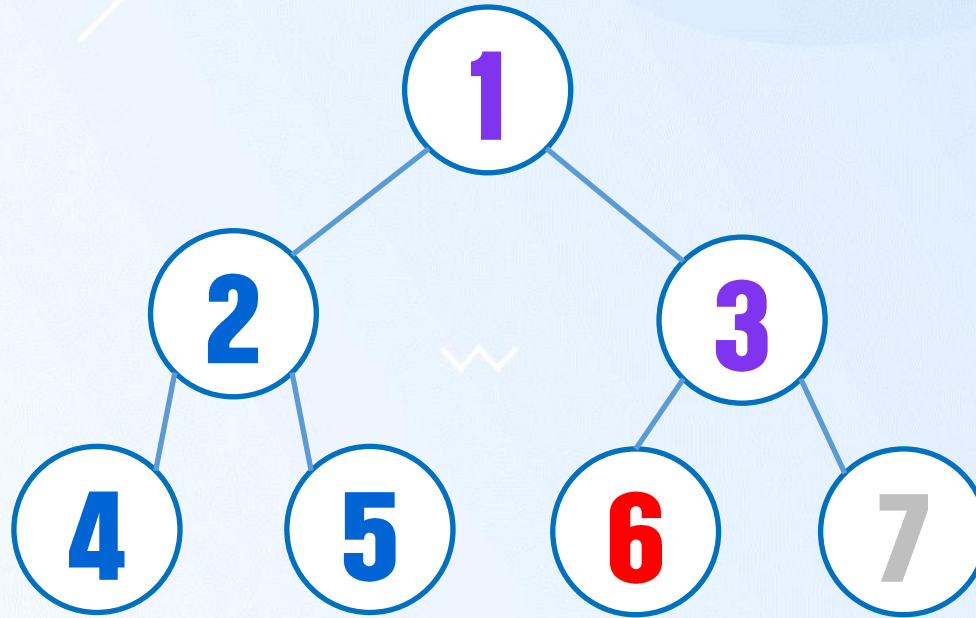
STACK

결과: 4 → 5 → 2

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=6  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       ✓ DFS(n * 2); → 12  
8         DFS(n * 2 + 1); → 13  
9         console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (6): 7에서 중지

D (3): 7에서 중지

D (1): 8에서 중지

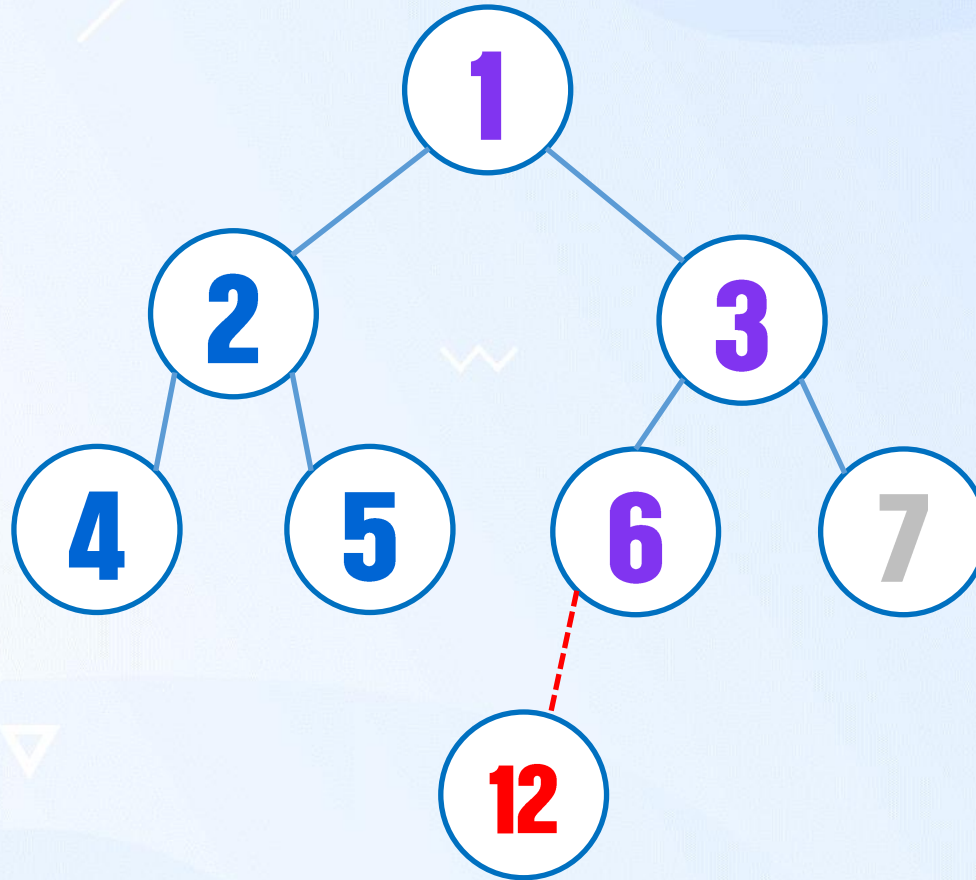
STACK

결과: 4 → 5 → 2

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=12  
3     // 🚧 종착 지점  
4     ✓ if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2);  
8       DFS(n * 2 + 1);  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (12) : pop

D (6) : 7에서 중지

D (3) : 7에서 중지

D (1) : 8에서 중지

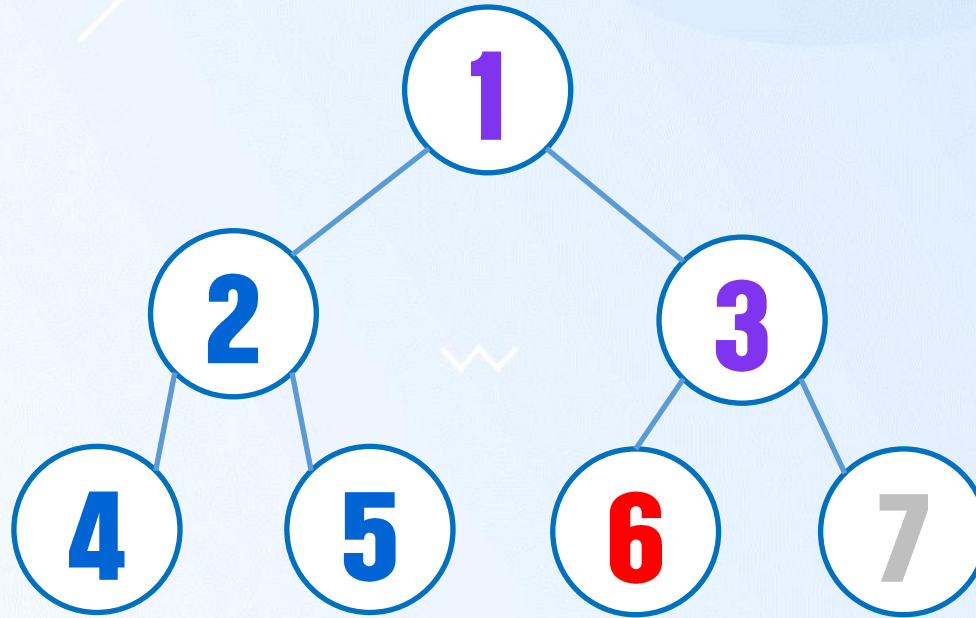
STACK

결과: 4 → 5 → 2

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=6  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 12  
8       ✓ DFS(n * 2 + 1); → 13  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (6) : 8에서 중지

D (3) : 7에서 중지

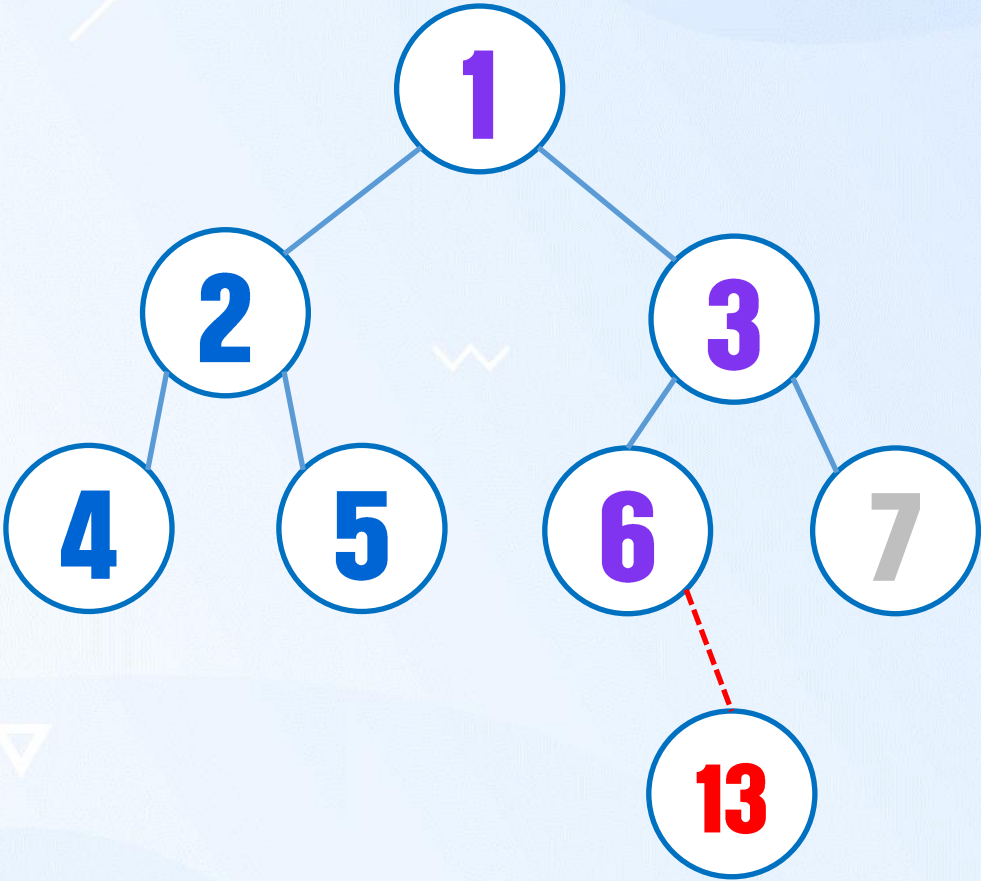
D (1) : 8에서 중지

STACK

결과: 4 → 5 → 2

이진트리 후위순회

```
1 function solution(n) {
2   function DFS(n) {      n=13
3     // 🚧 종착 지점
4     ✓ if (n > 7) return;
5     else {
6       // 🚧 계속 트리가 뿔어나가는 지점
7       DFS(n * 2);
8       DFS(n * 2 + 1);
9       console.log(n); // 후위 순회
10    }
11  }
12  DFS(n);
13 }
14 solution(1);
```



D (12) : pop
D (6) : 7에서 중지
D (3) : 7에서 중지
D (1) : 8에서 중지

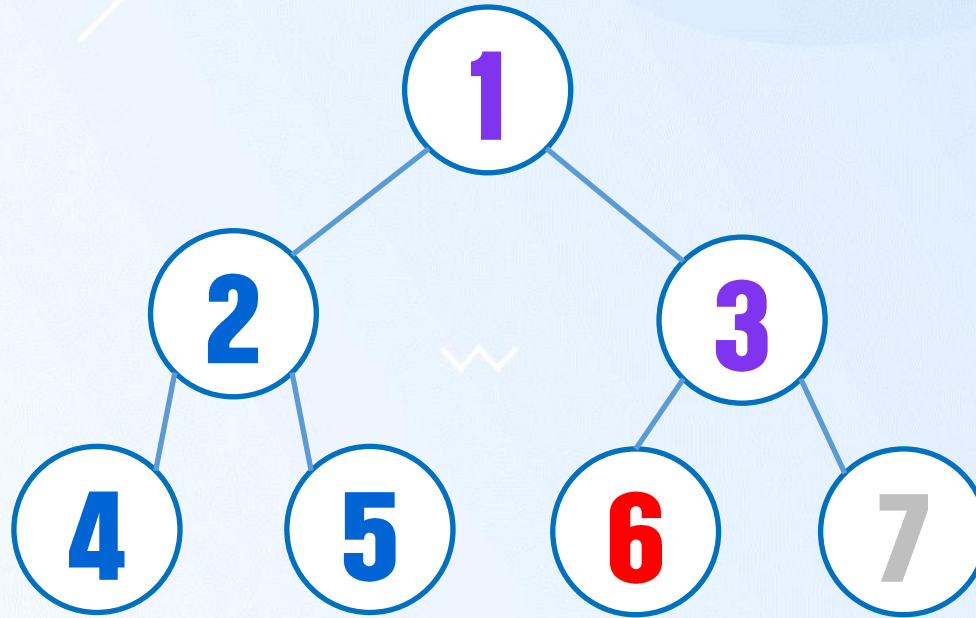
STACK

결과: 4 → 5 → 2

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=6  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 12  
8       DFS(n * 2 + 1); → 13  
9       ✓ console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (6) : pop

D (3) : 7에서 중지

D (1) : 8에서 중지

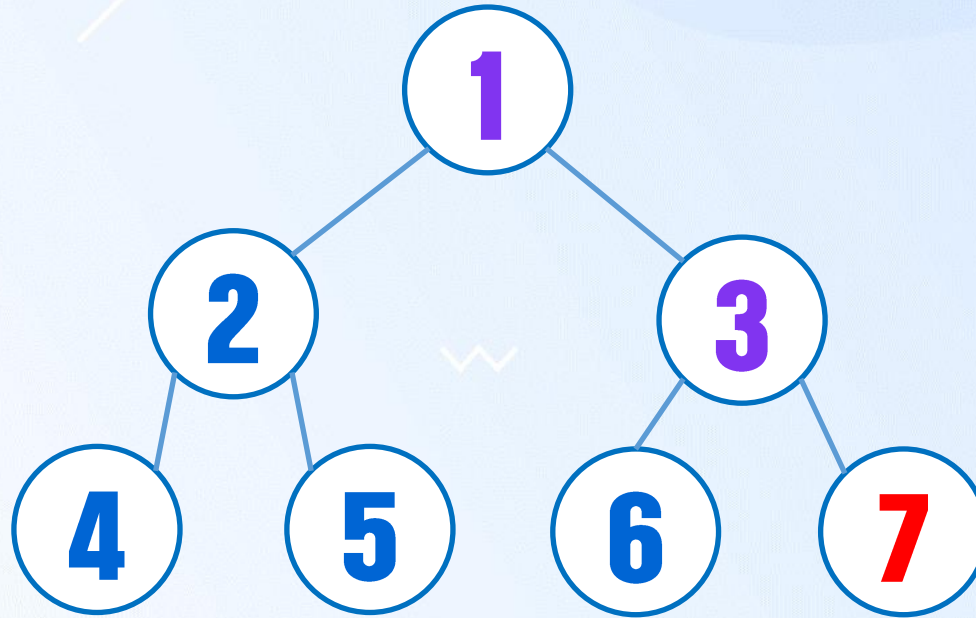
STACK

결과: 4 → 5 → 2 → 6

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=7  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       ✓ DFS(n * 2); → 14  
8         DFS(n * 2 + 1); → 15  
9         console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (7): 7에서 중지

D (3): 7에서 중지

D (1): 8에서 중지

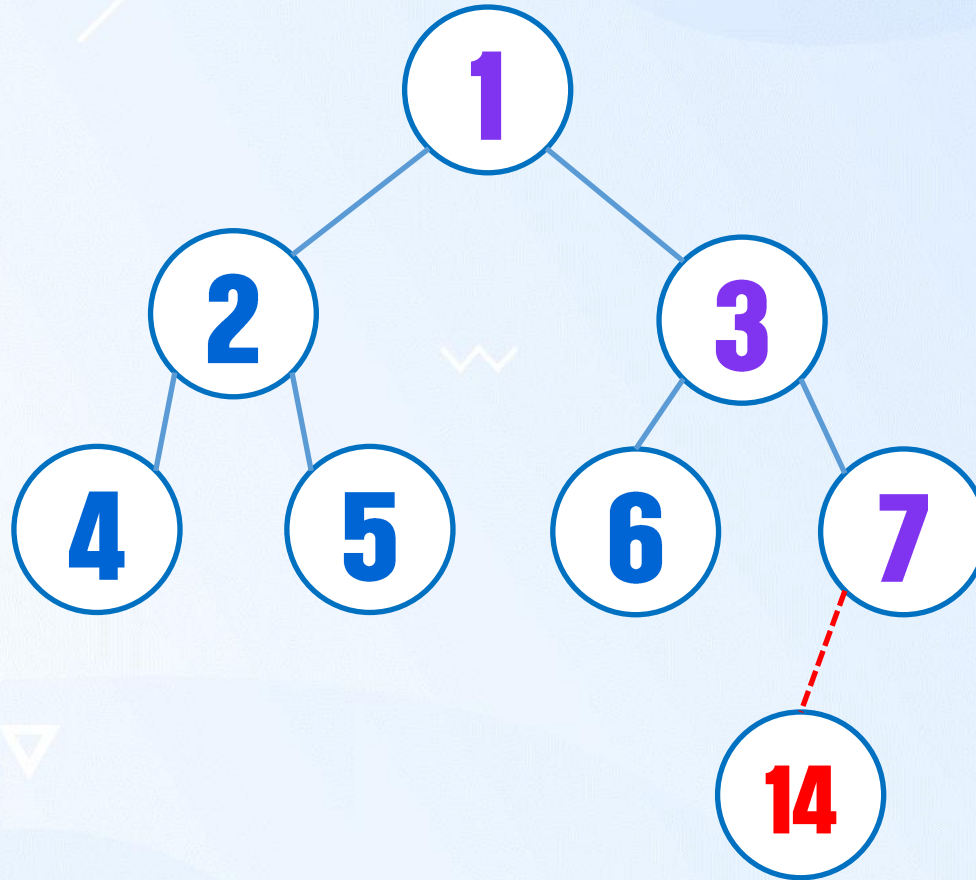
STACK

결과: 4 → 5 → 2 → 6

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=14  
3     // 🚧 종착 지점  
4     ✓ if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2);  
8       DFS(n * 2 + 1);  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (14) : pop

D (7) : 7에서 중지

D (3) : 7에서 중지

D (1) : 8에서 중지

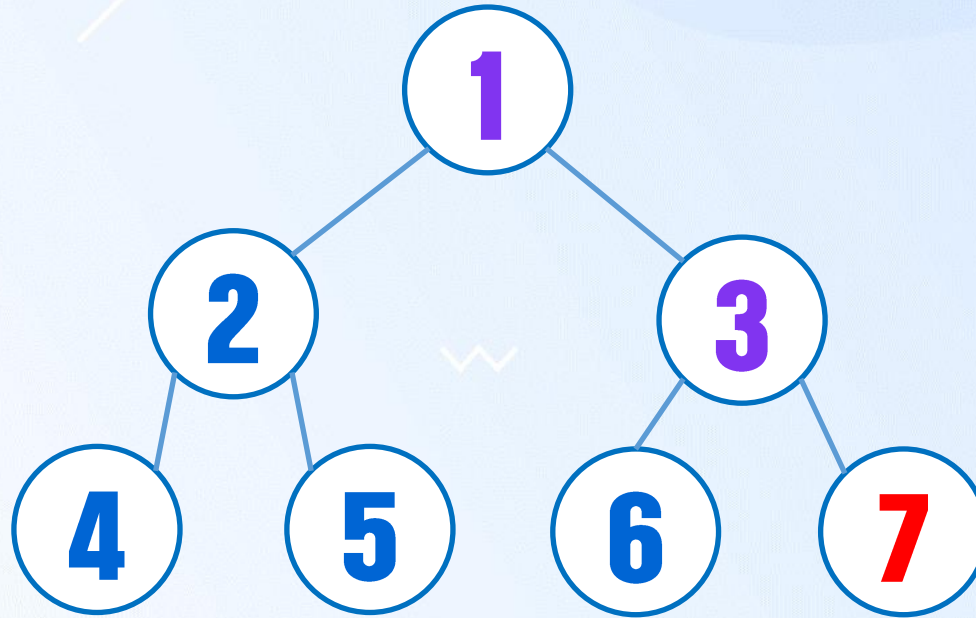
STACK

결과: 4 → 5 → 2 → 6

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=7  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 14  
8       ✓ DFS(n * 2 + 1); → 15  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (7): 8에서 중지

D (3): 7에서 중지

D (1): 8에서 중지

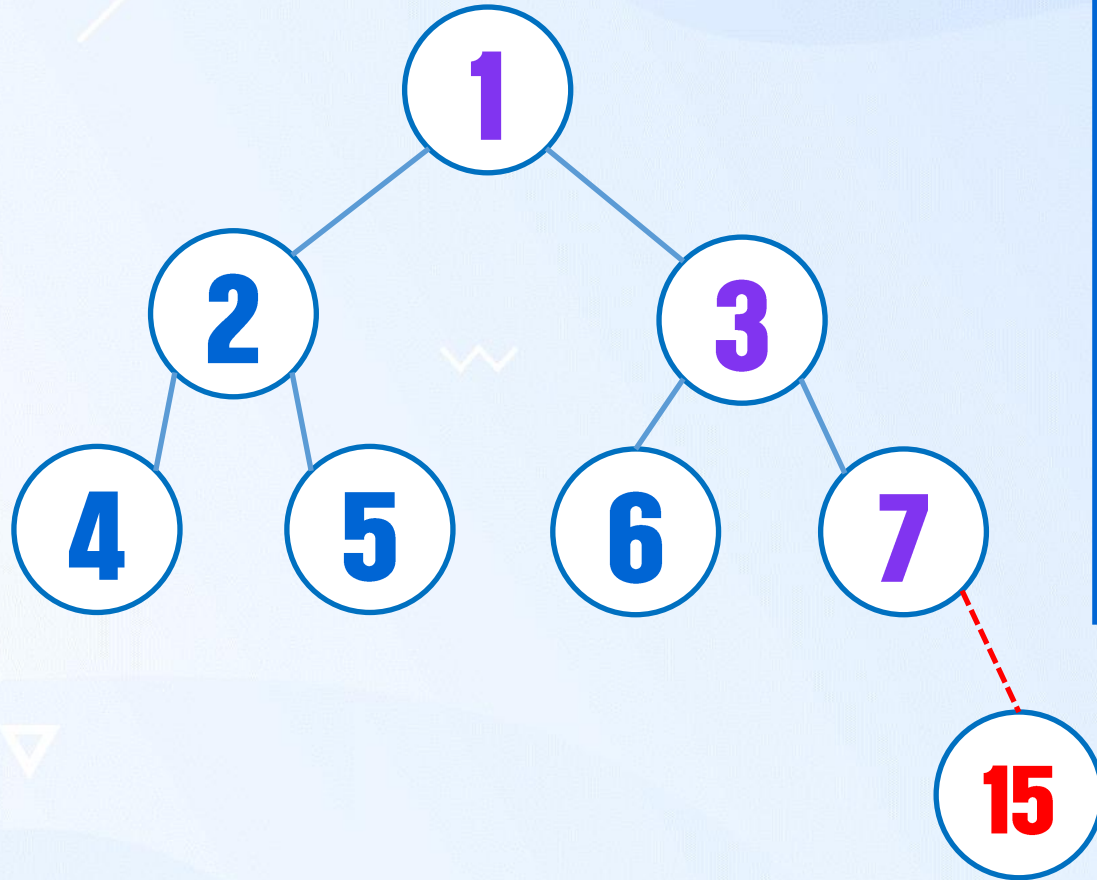
STACK

결과: 4 → 5 → 2 → 6

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=15  
3     // 🚧 종착 지점  
4     ✓ if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2);  
8       DFS(n * 2 + 1);  
9       console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



D (15): pop
D (7): 8에서 중지
D (3): 7에서 중지
D (1): 8에서 중지

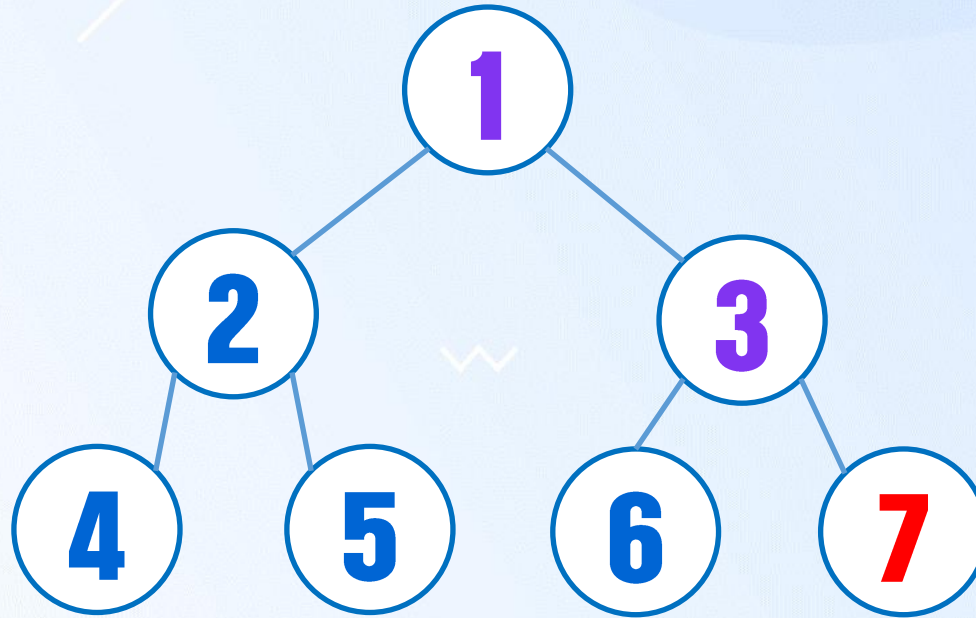
STACK

결과: 4 → 5 → 2 → 6

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=7  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뻗어나가는 지점  
7       DFS(n * 2); → 14  
8       DFS(n * 2 + 1); → 15  
9       ✓ console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



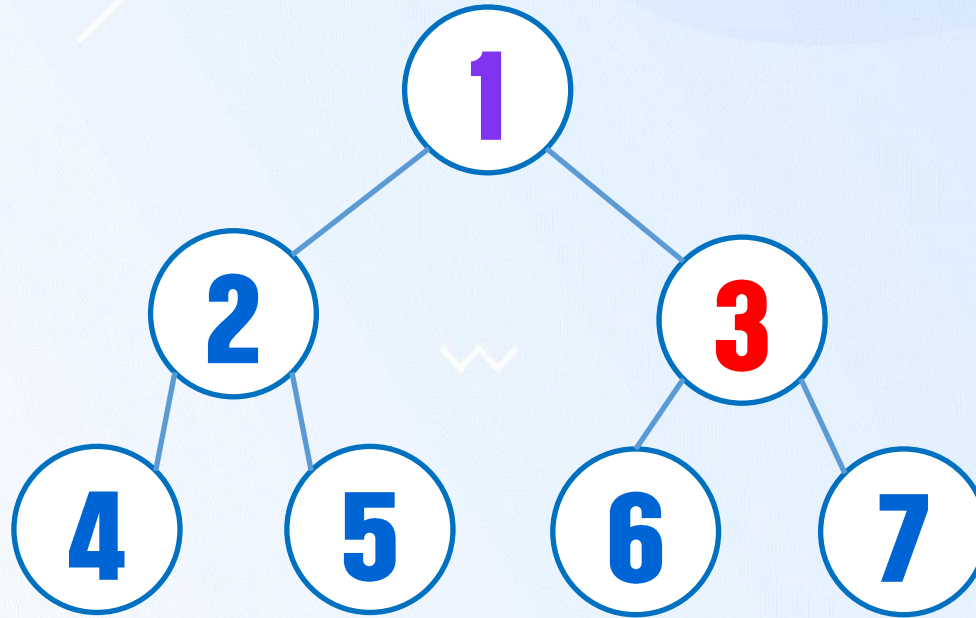
STACK

결과: 4 → 5 → 2 → 6 → 7

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=3  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 6  
8       DFS(n * 2 + 1); → 7  
9       ✓ console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



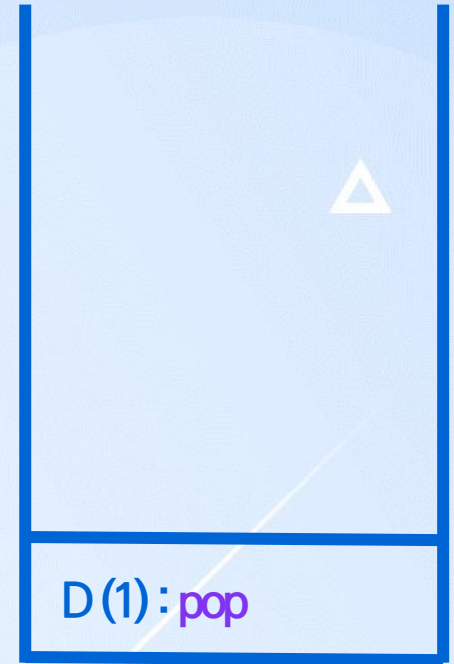
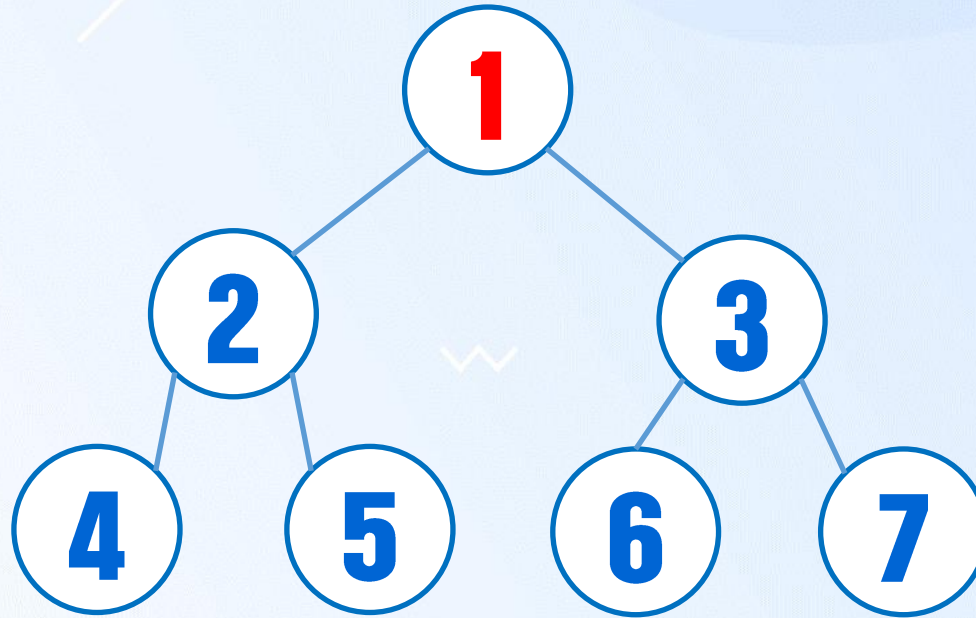
STACK

결과: 4 → 5 → 2 → 6 → 7 → 3

이진트리 후위순회



```
1 function solution(n) {  
2   function DFS(n) {    n=1  
3     // 🚧 종착 지점  
4     if (n > 7) return;  
5     else {  
6       // 🚧 계속 트리가 뿔어나가는 지점  
7       DFS(n * 2); → 2  
8       DFS(n * 2 + 1); → 3  
9       ✓ console.log(n); // 후위 순회  
10    }  
11  }  
12  DFS(n);  
13 }  
14 solution(1);
```



STACK

결과: 4 → 5 → 2 → 6 → 7 → 3 → 1

감사합니다

