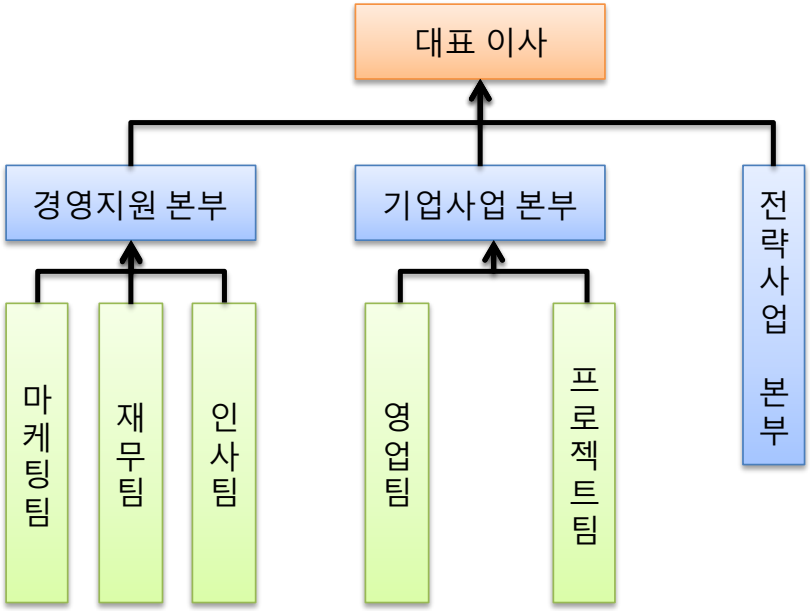
트리

목차

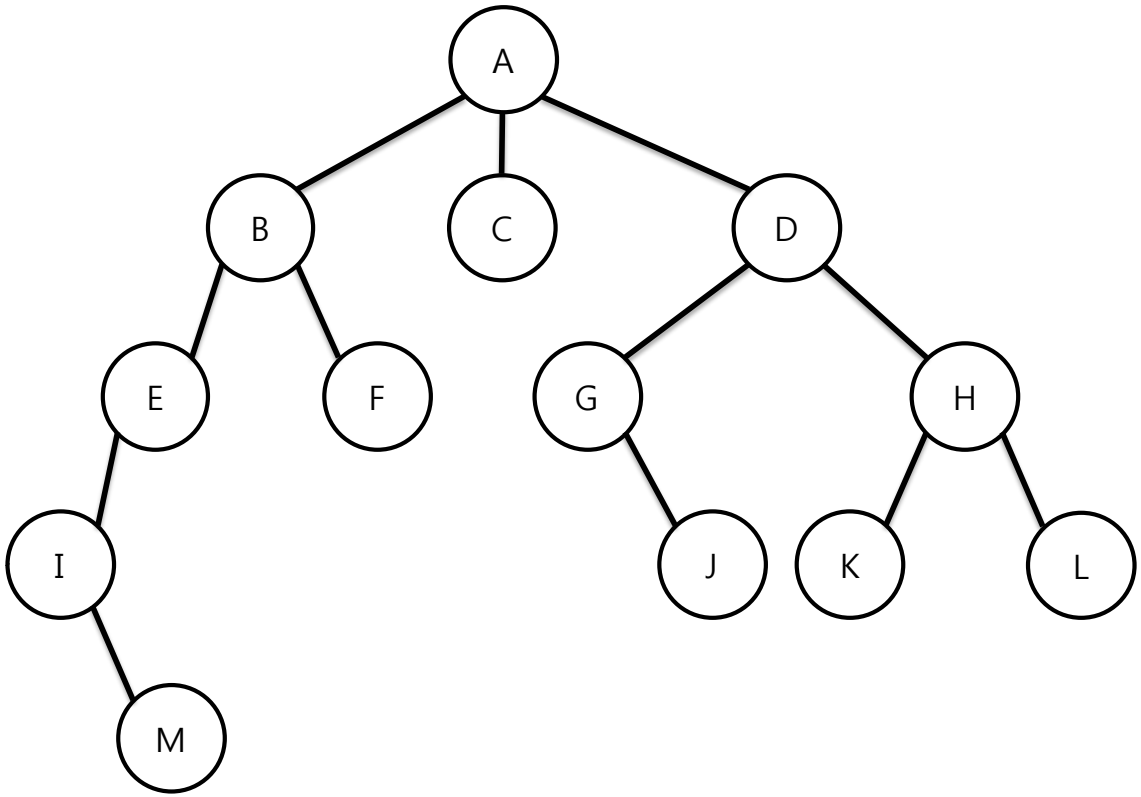
1. 트리의 개념 (1/5)

- 트리(Tree) = 노드(Node) + 간선(Edge)
- 계층 구조(Hierarchical Structure)
  - 부모-자식(Parent-Child) 관계
    - Cf) 1:N 관계
  - 응용 어플리케이션, 알고리즘



1. 트리의 개념 (2/5)

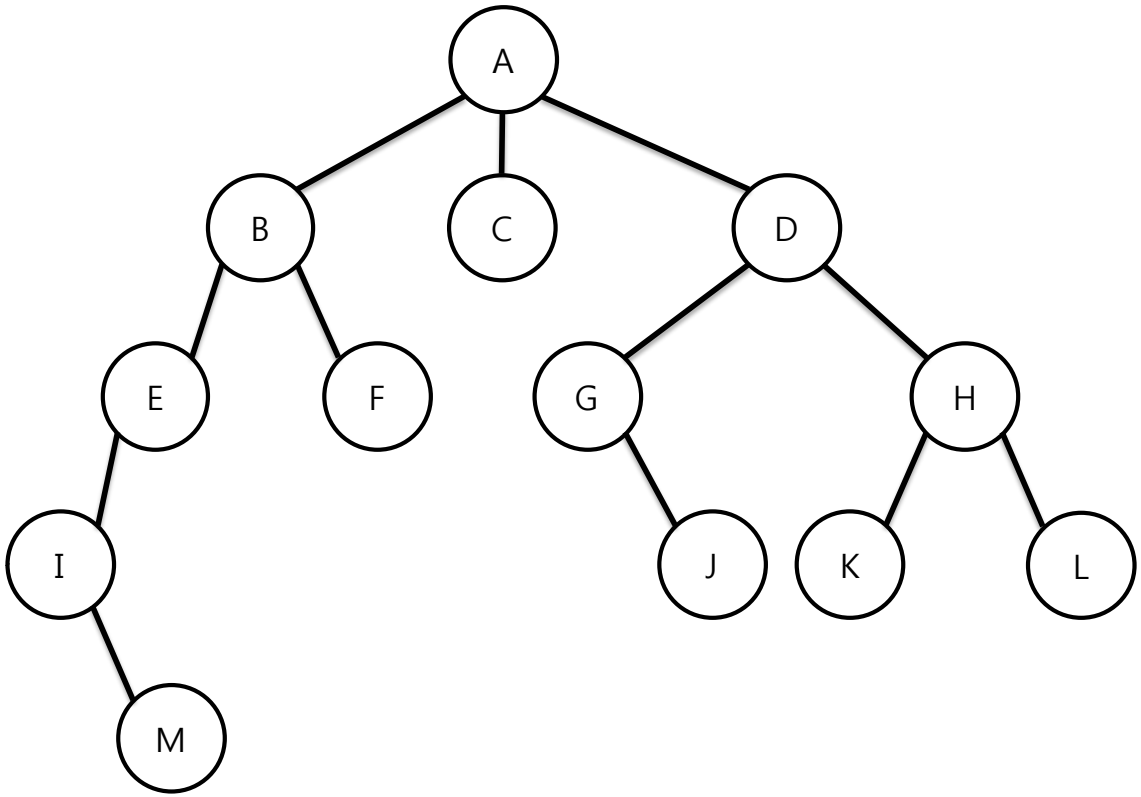
- 용어들 (노드 종류, 트리에서의 위치)
  - 루트(Root) 노드
  - 단말(Leaf 혹은 Terminal) 노드
    - 자식 노드가 없는 경우
  - 내부(Internal) 노드



1. 트리의 개념 (3/5)

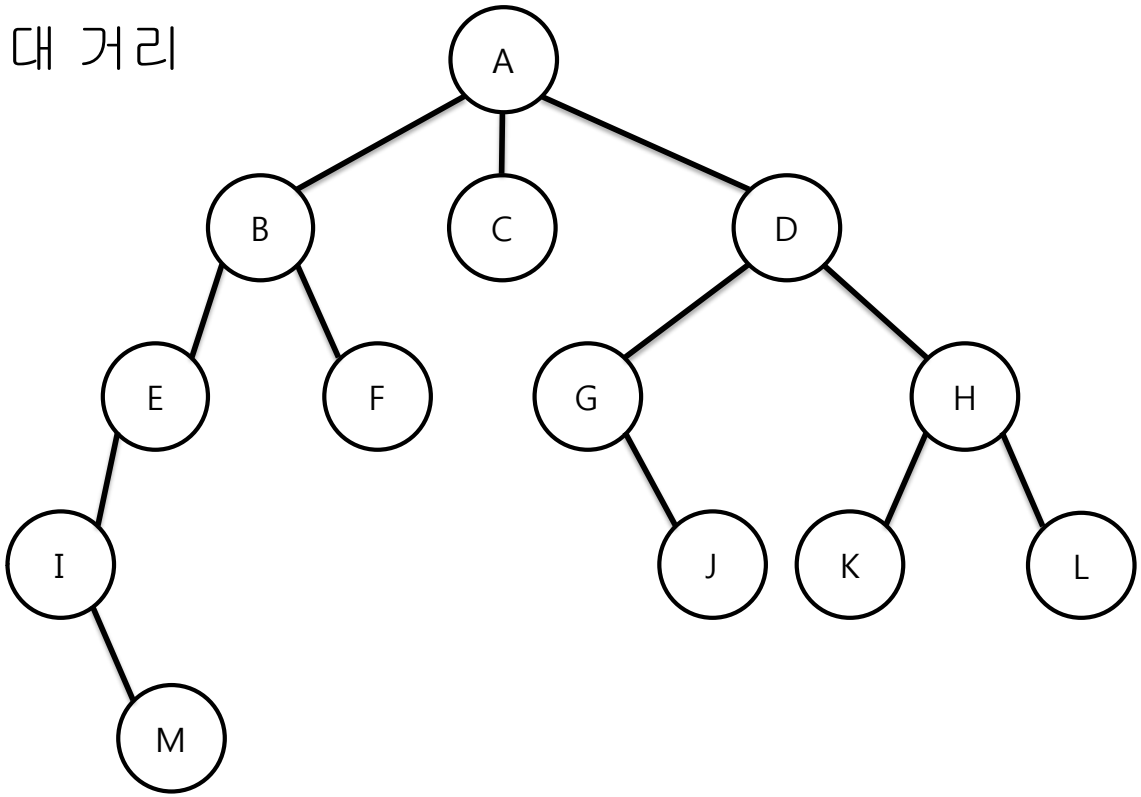
- 용어들(계속, 노드 사이의 관계 관점)
  - 부모(Parent) 노드
  - 자식(Child) 노드
  - 선조(Ancestor) 노드
  - 후손(Descendant) 노드
  - 형제(Sibling) 노드

  - E.g. 노드 E의 경우



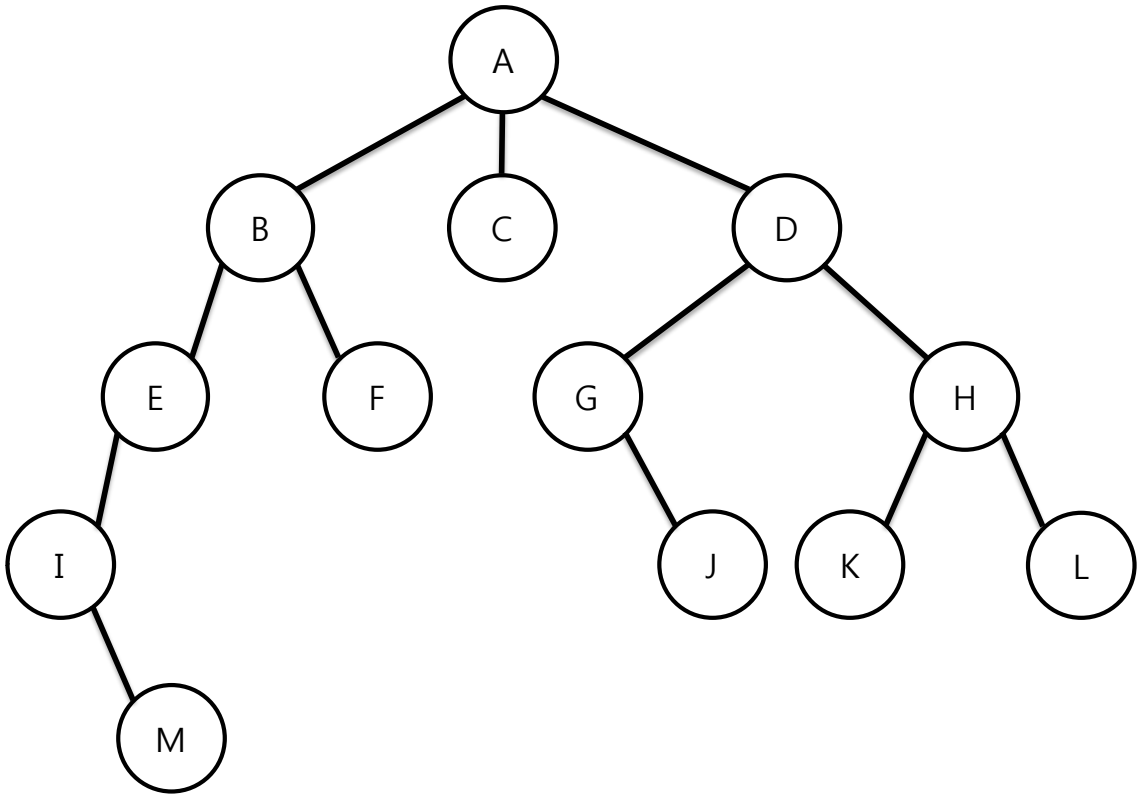
1. 트리의 개념 (4/5)

- 용어들 (계속, 속성 관점)
  - 레벨(Level)
    - 루트 노드로부터의 거리
      - Cf) 루트노드는 1
  - 높이(Height)
    - 단말 노드로부터의 최대 거리
      - Cf) 단말 노드는 1
  - 차수(Degree)
    - 자식 노드의 개수



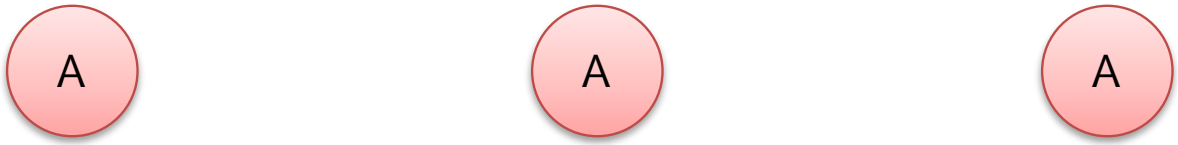
1. 트리의 개념 (5/5)

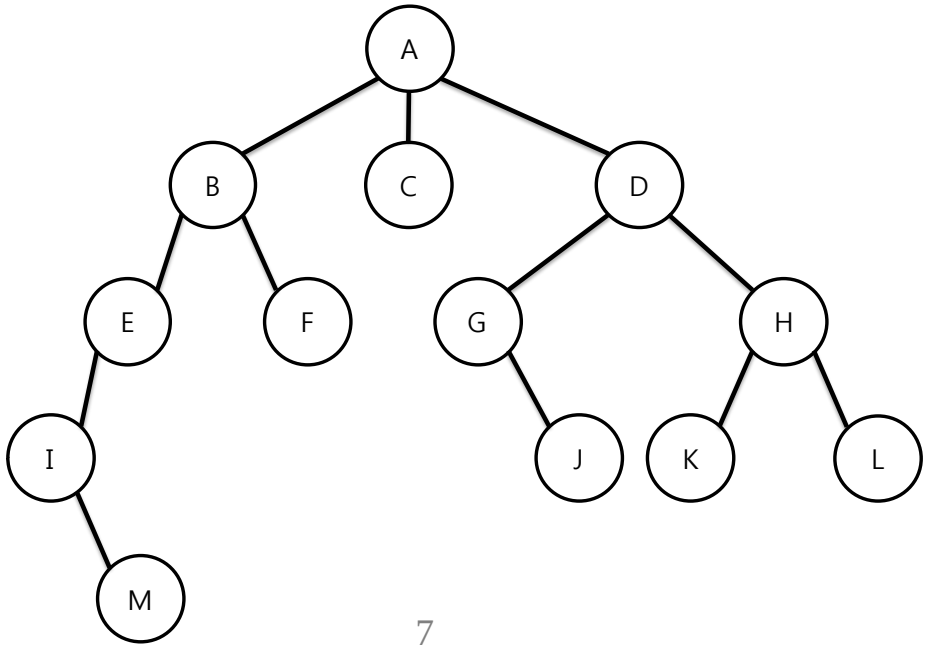
- 그 밖에
  - 서브트리(Subtree)
    - 전체 트리의 부분집합
  - 포리스트(Forest)
    - 트리들의 집합



2. 이진 트리 (1/4)

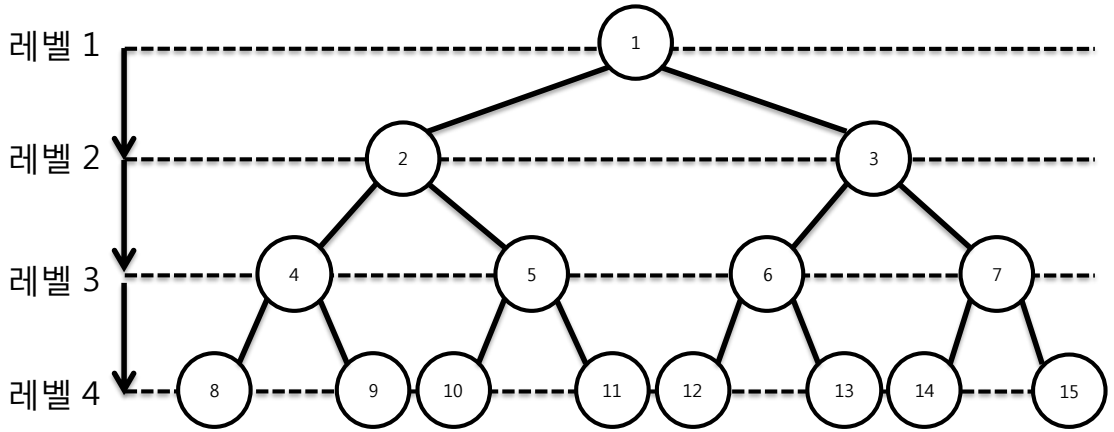
- 이진 트리(Binary Tree)
  - 가장 간단한 구조를 가지는 트리
  - 다양한 알고리즘에 사용
  - 최대 차수 = 2





2. 이진 트리 (2/4)

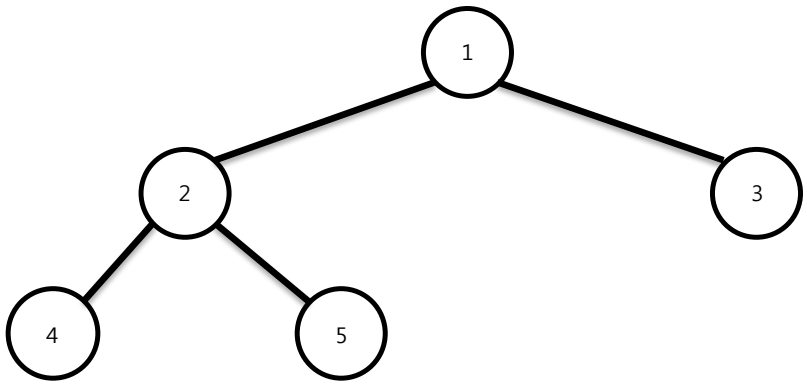
- 이진 트리의 종류 (알고리즘 성능)
  - 포화 이진 트리

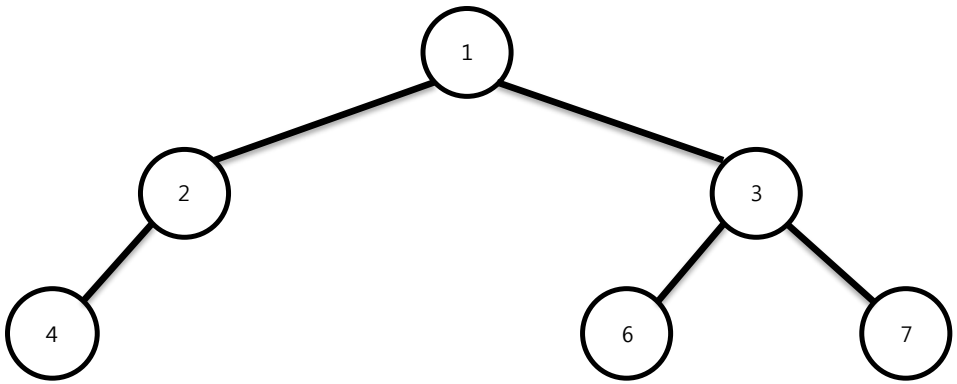


  - 레벨 h의 경우 노드의 개수?

2. 이진 트리 (3/4)

- 이진 트리의 종류(계속)
  - 완전 이진 트리
    - 히프에서 사용
    - 레벨이 h의 경우
      - 레벨 h-1까지는 포화 이진 트리

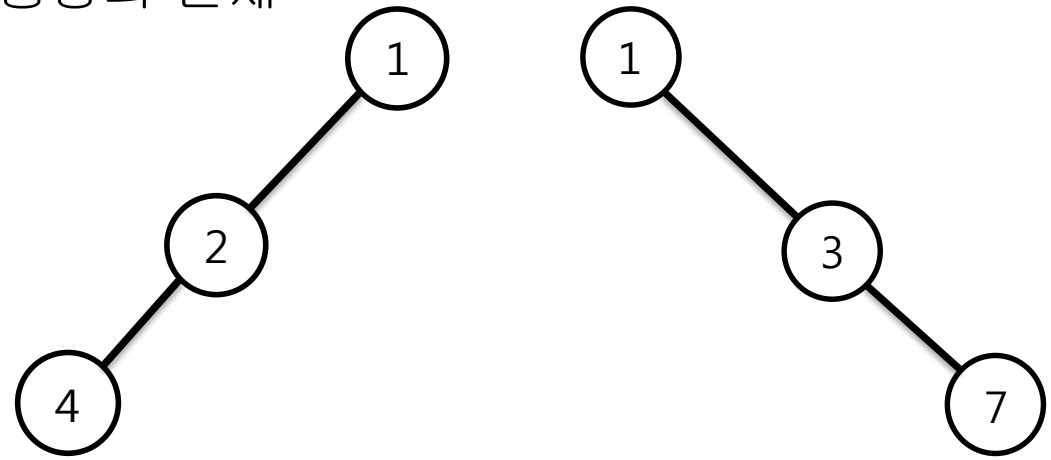




      - 레벨 h의 경우: 왼쪽부터 차례대로

2. 이진 트리 (4/4)

- 이진 트리의 종류 (계속)
  - 편향 이진 트리
    - 노드 개수 n = h
    - 이진탐색트리
      - 성능상의 문제

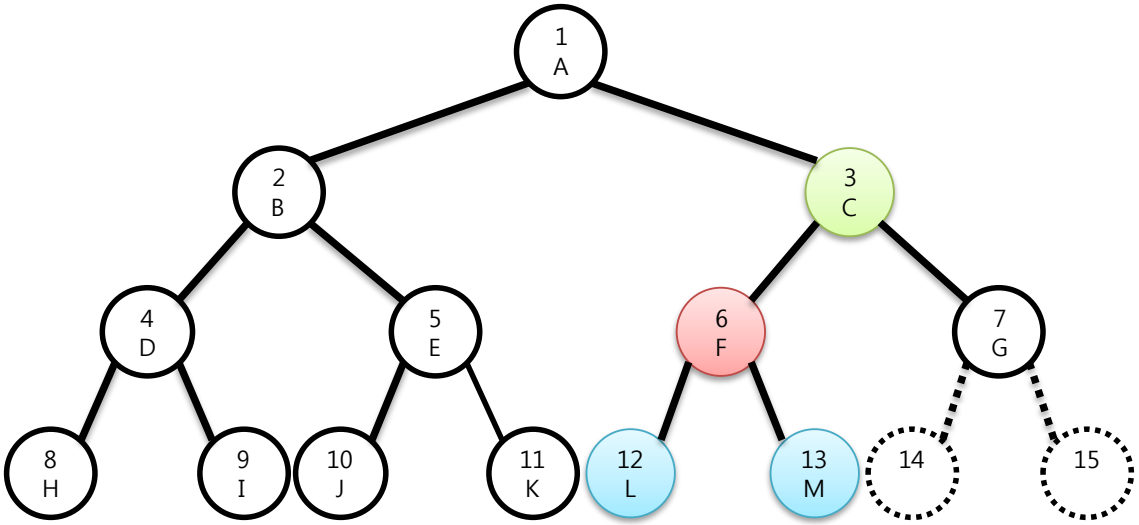


2.2 이진 트리의 추상 자료형

- 이진 트리의 추상 자료형
  - 이진 트리 생성
  - 이진 트리 삭제
  - 루트 노드 반환
  - 왼쪽 자식 노드 추가
  - 오른쪽 자식 노드 추가
  - 왼쪽 자식 노드 반환
  - 오른쪽 자식 노드 반환

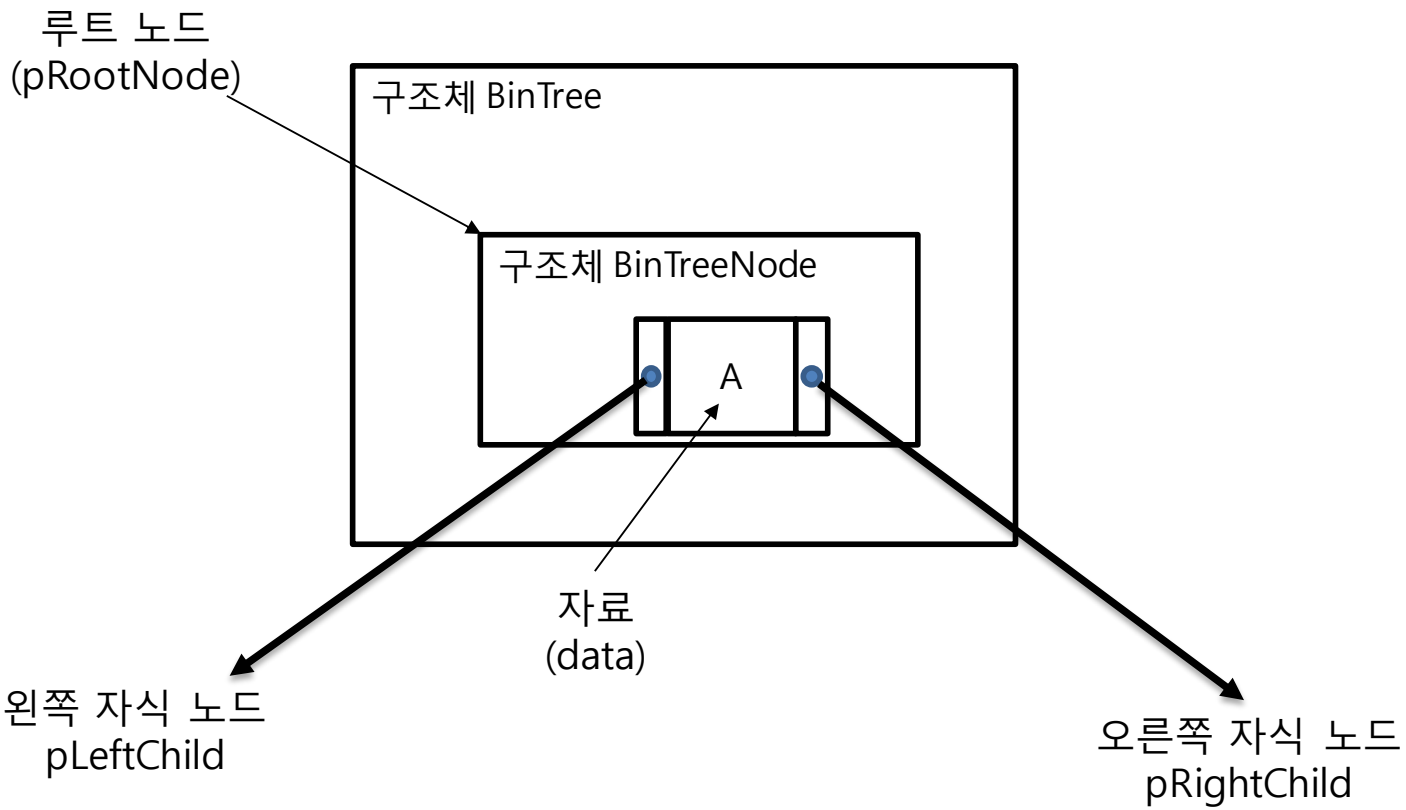
2.3 배열을 이용한 이진 트리 구현

- 노드 번호 / 저장 자료
  - 노드 i
    - 부모 노드 인덱스 = ⌊i/2⌋ , 단, i > 1
    - 왼쪽 자식 노드 인덱스 = 2 x I
    - 오른쪽 자식 노드 인덱스 = (2 x i) + 1



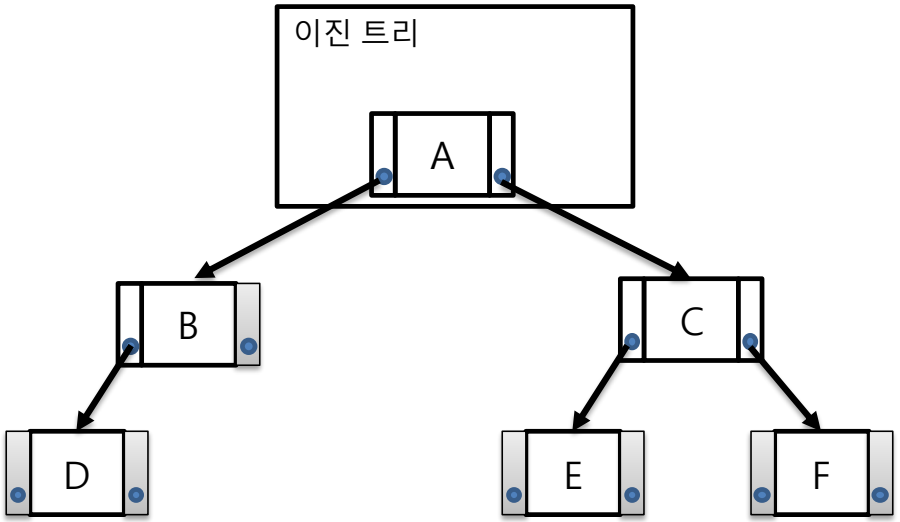
0	
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	
15	

2.4. 포인터를 이용한 이진 트리 구현 (2/3)



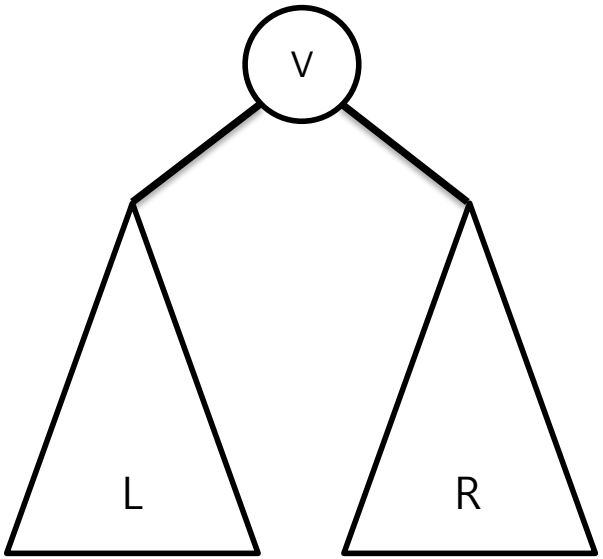
2.4. 포인터를 이용한 이진 트리 구현 (3/3)

- 이진 트리의 생성
- 이진 트리의 자식 노드 추가
- 이진 트리의 자식 노드 반환
- 이진 트리 삭제
- 예제 프로그램



3. 이진 트리의 순회 (1/11)

- 순회(traversal)

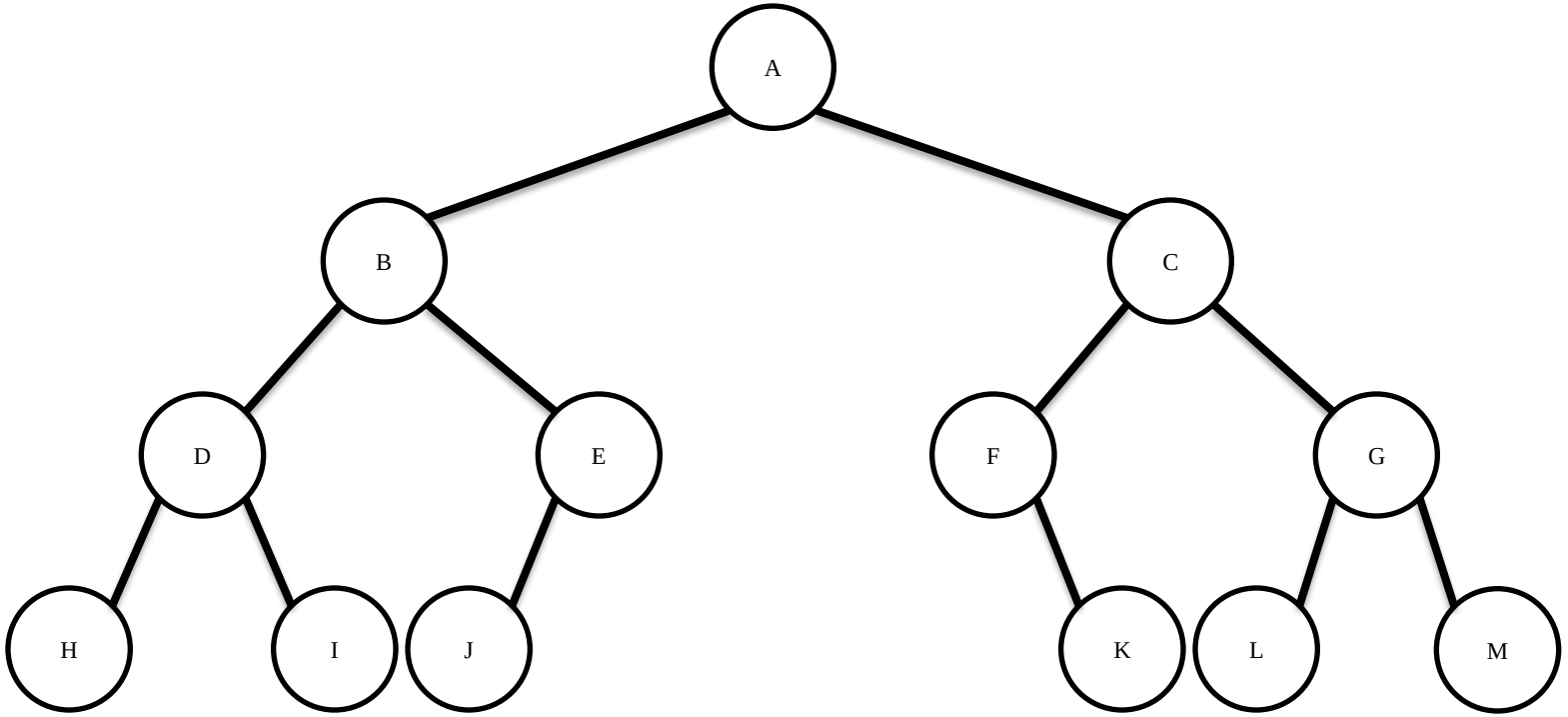


  - 전위 순회(Preorder Traversal)
  - 중위 순회(Inorder Traversal)
  - 후위 순회(Postorder Traversal)

※ 레벨 순회(Level Traversal)

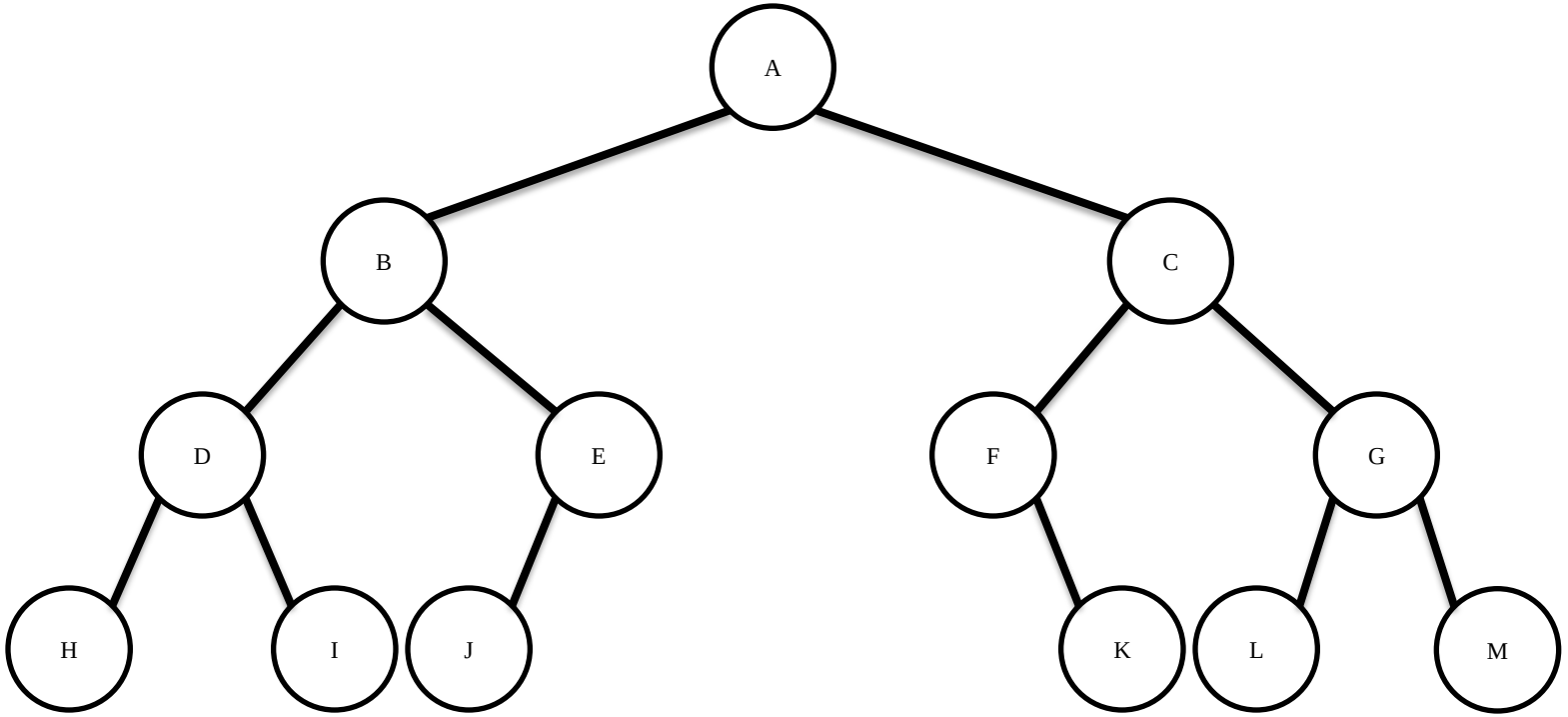
3. 이진 트리의 순회 (2/11)

- 전위 순회
  - 방문 순서: V-L-R



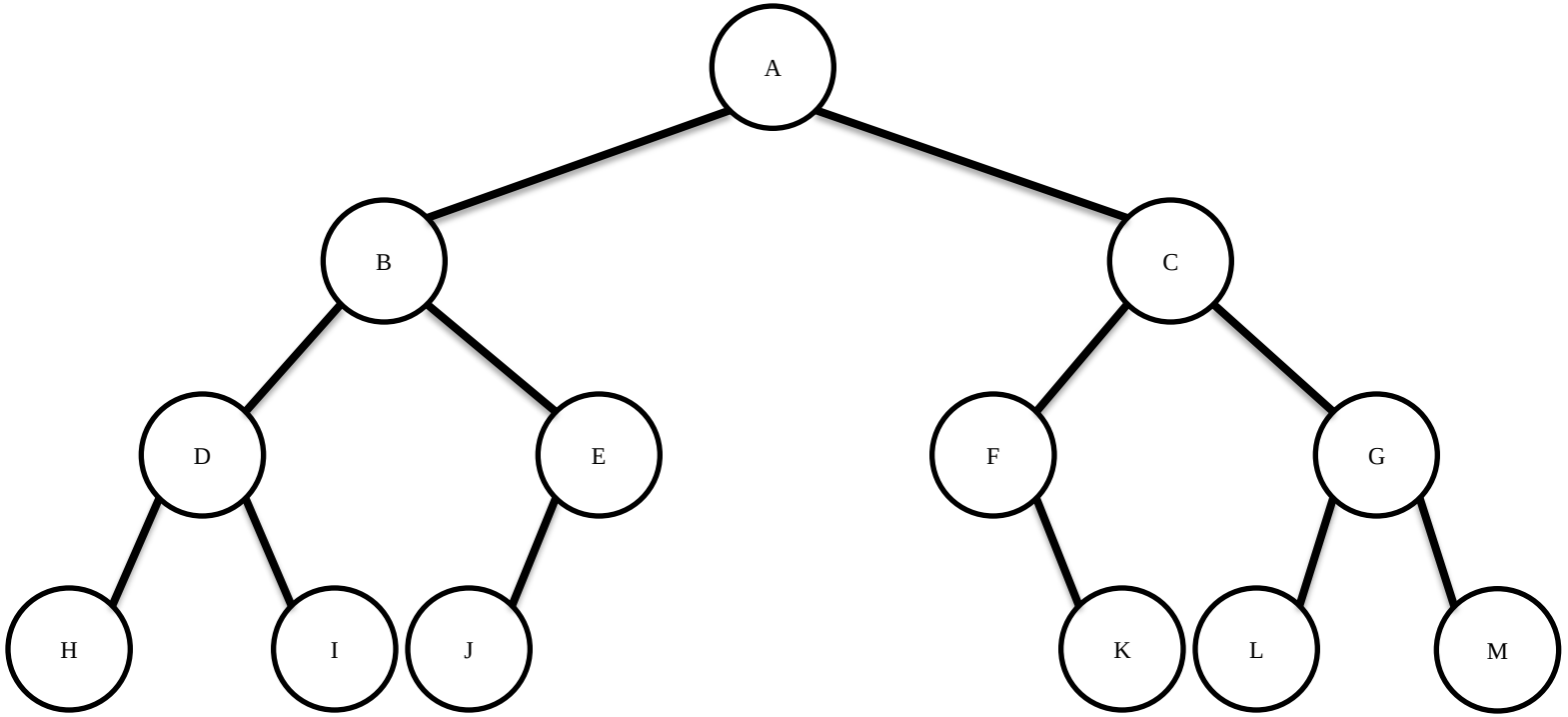
3. 이진 트리의 순회 (3/11)

- 중위 순회
  - 방문 순서: L-V-R



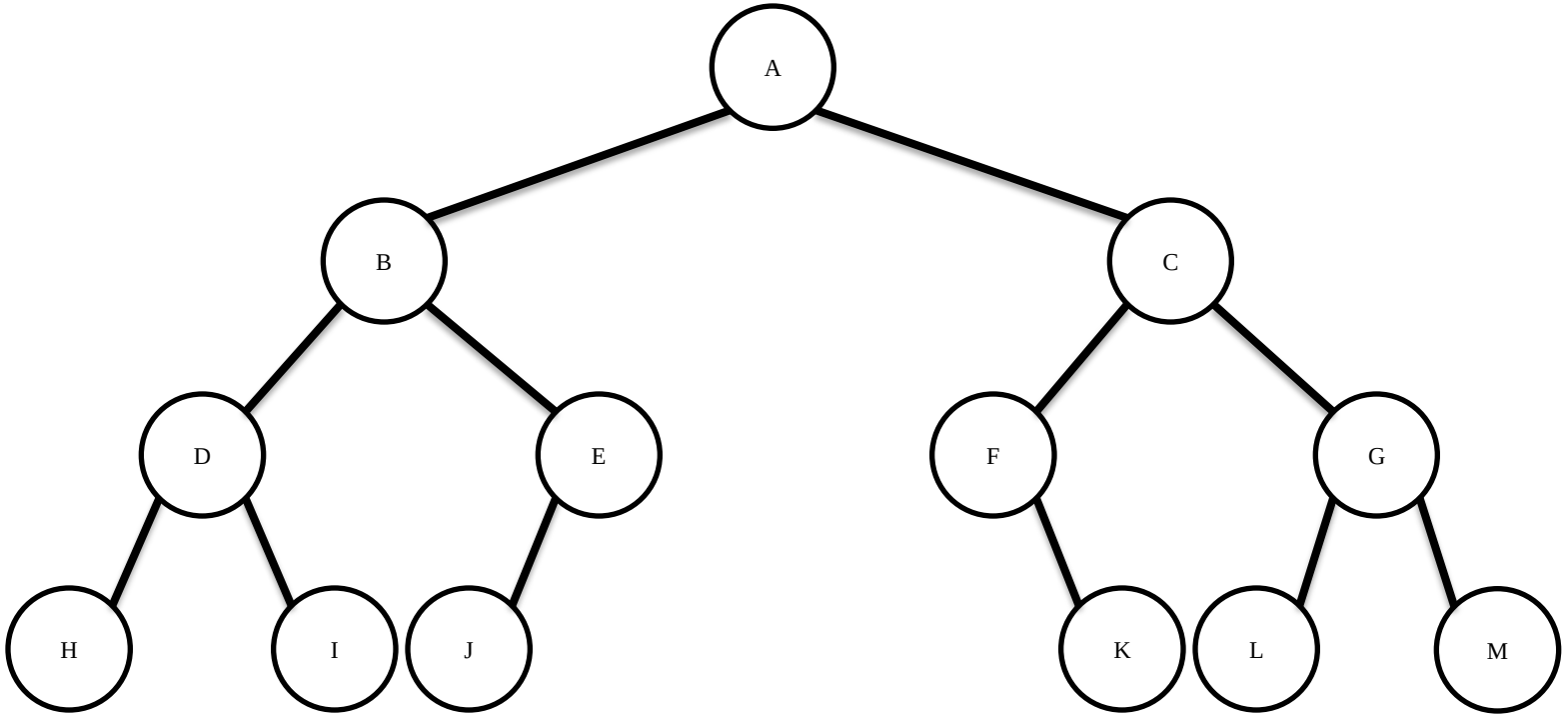
3. 이진 트리의 순회 (4/11)

- 후위 순회
  - 방문 순서: L-R-V



3 이진 트리의 순회 (5/11)

- 레벨 순회



3. 이진 트리의 순회 (6/11)

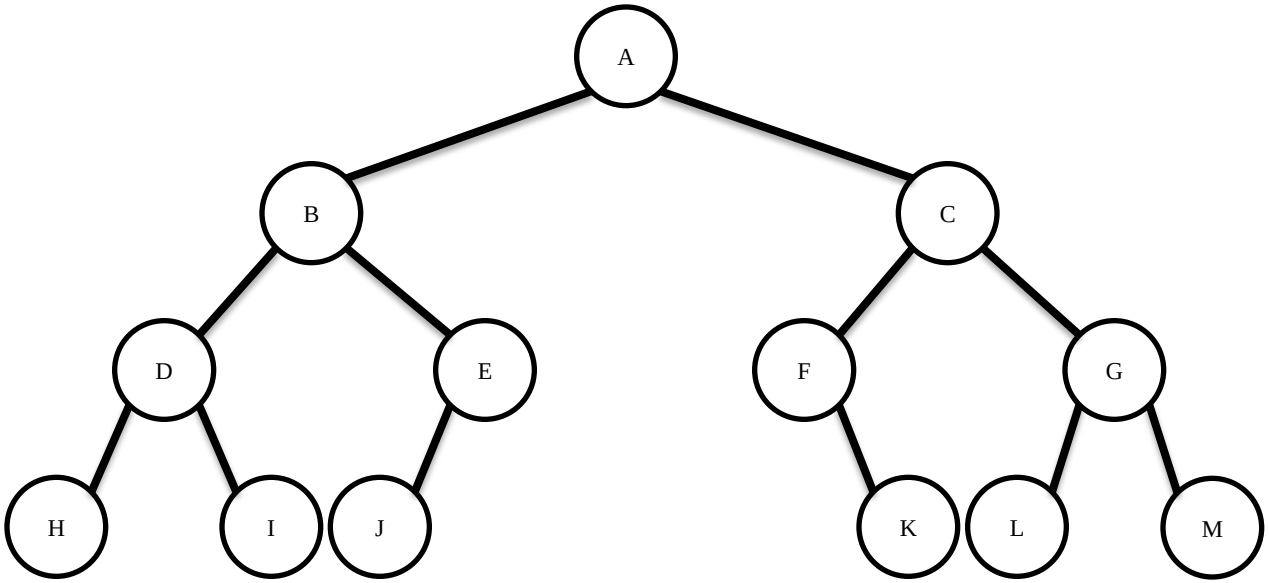
- 구현 I: 재귀 호출에 의한 구현

※ 포인터를 이용한 이진 트리 구현 참고

3. 이진 트리의 순회 (7/11)

- 구현 I: 재귀 호출에 의한 구현 (계속)

구분	순서
전위 순회	A B D H I E J C F K G L M
중위 순회	H D I B J E A F K C L G M
후위 순회	H I D J E B K F L M G C A



3. 이진 트리의 순회 (8/11)

- 구현 II: 반복 호출에 의한 구현

※ 2.4. **포인터를 이용한 이진 트리 구현 참고**
4.4. **연결 리스트로 구현한 스택 참고**
5.5. **연결 리스트로 구현한 큐 참고**

3. 이진 트리의 순회 (11/11)

- 반복 전위 순회
  - preorderTraversalBinTree()

- 반복 중위 순회
  - inorderTraversalBinTree()

- 반복 레벨 순회
  - levelOrderTraversalBinTree()

4. 이진 트리의 연산 (1/2)

- 이진 트리 관련 함수들
  - 이진 트리 복사
  - 이진 트리 동질성 검사
  - 이진 트리 노드 개수 구하기
  - 이진 트리의 단말 노드 개수 구하기
  - 이진 트리의 높이 구하기
  - 이진 트리 구조 및 내용 출력하기

4. 이진 트리의 연산 (2/2)

- 이진 트리 복사
  - 재귀 전위 순회 방식 이용

- 이진 트리 동질성 검사
  - 구조 및 저장된 자료 검사
  - 재귀 전위 순회 방식 이용

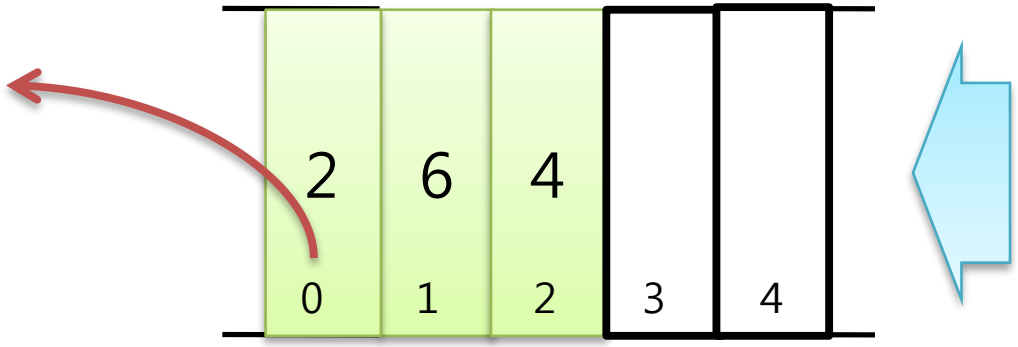
- 이진 트리 노드 개수 구하기
  - 재귀 후위 순회 방식 이용
- 이진 트리의 단말 노드 개수 구하기

- 이진 트리의 높이 구하기

- 이진 트리 구조 및 내용 출력하기

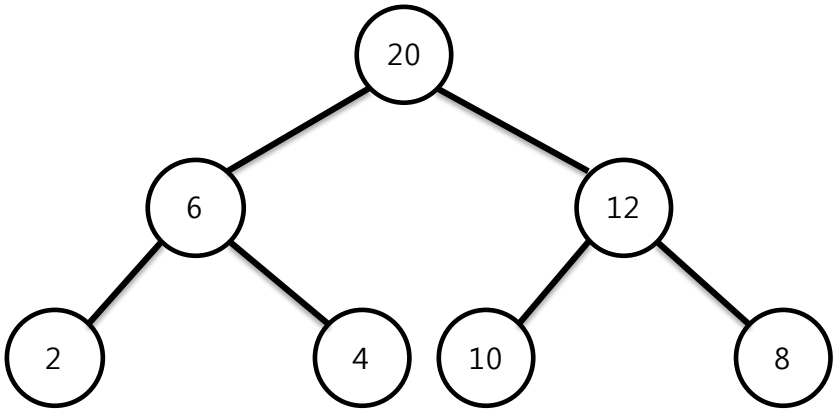
5. 히프 (1/9)

- (일반)큐와 우선순위 큐

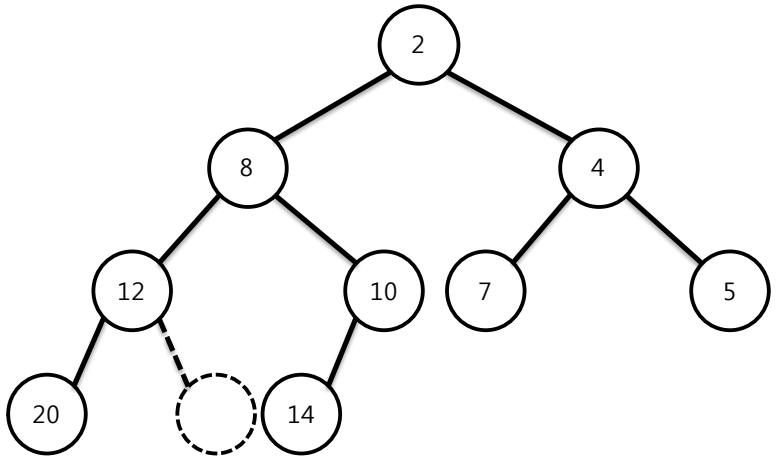


5. 히프 (2/9)

- 최대 히프 (Max Heap)
  - 최대 트리 + 완전 이진 트리
    - 최대 트리
      - 루트노드 의 값: 최댓값
      - 부모 노드의 키 값 >= 자식 노드의 키 값
  - 최대 우선순위 큐

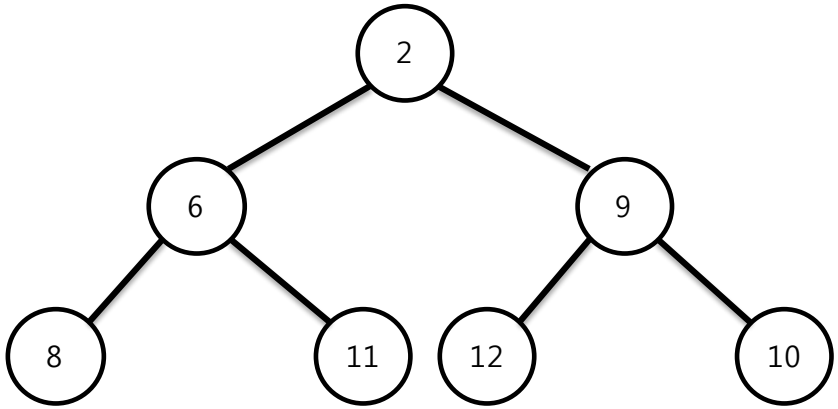


※ **완전이진트리**



5. 히프 (3/9)

- 최소 히프 (Min Heap): 최소 우선순위 큐
  - 최소 트리 + 완전 이진 트리
  - 최소 트리
    - 루트 노드 값은 최솟값
    - 부모 노드 값 <= 자식 노드의 값



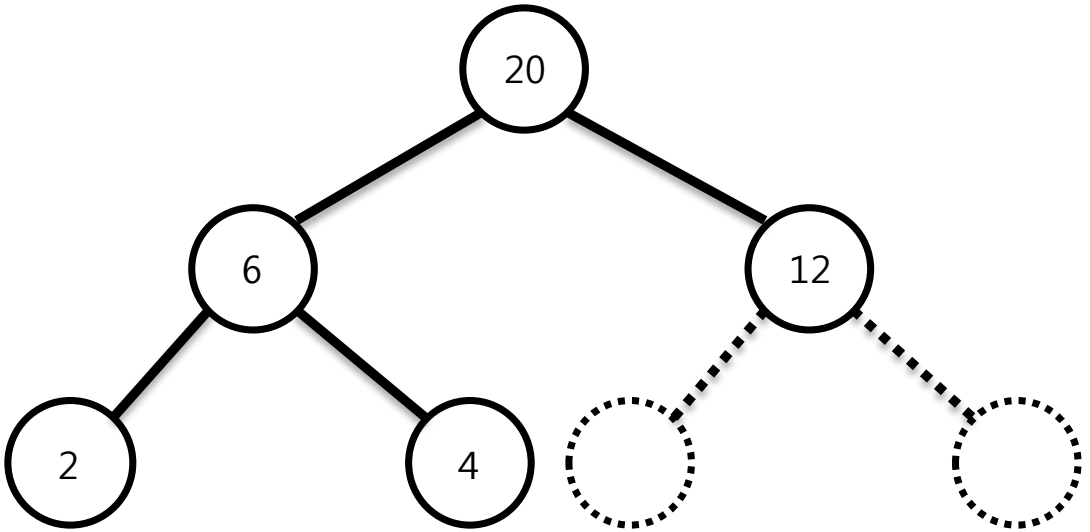
5. 히프 (4/9)

- 추상 자료형
  - 히프 생성
  - 히프 삭제
  - 자료 추가
  - 자료 제거

5. 히프 (5/9)

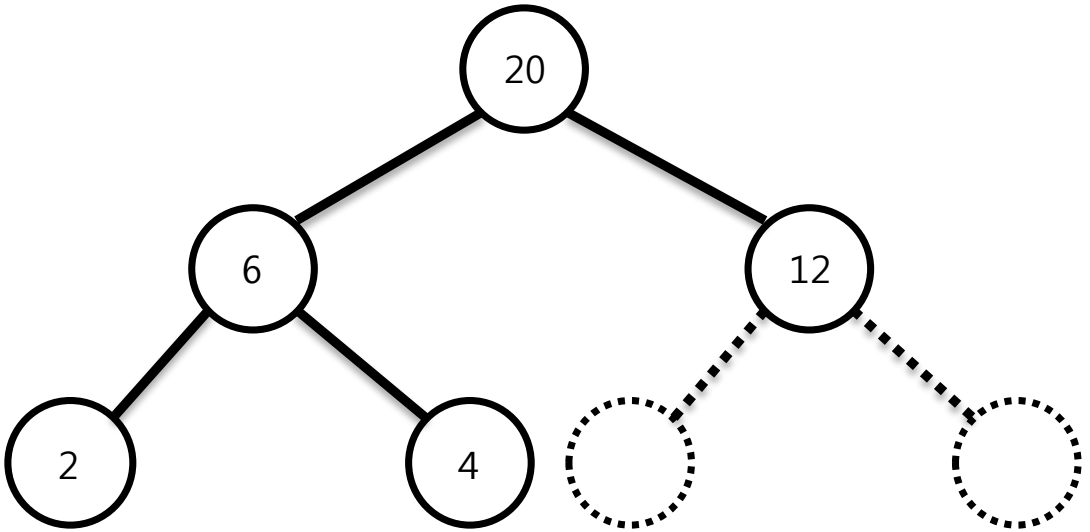
- 최대 히프의 자료 추가 연산
  - Step-1) 트리의 마지막 자리에 임시 저장
  - Step-2) 부모 노드와 키 값 비교 및 이동

- 예)
  - 30의 추가



5. 히프 (6/9)

- 최대 히프의 삭제 연산
  - Step-1) 루트 노드의 삭제
  - Step-2) 트리 마지막 자리 노드의 임시 이동
  - Step-3) 자식 노드와 키 값 비교 및 이동



5. 히프 (8/9)

- 히프의 구현
  - 배열을 이용한 이진 트리 구현
    - 노드 i의 부모 노드 인덱스 = ⌊i/2⌋ , 단, i > 1
    - 노드 i의 왼쪽 자식 노드 인덱스 = 2 x i
    - 노드 i의 오른쪽 자식 노드 인덱스 = (2 x i) + 1

```
insertMaxHeap ( heap, data ) {
        // Step-1) 트리의 마지막 자리에 임시 저장
        heap->currentElementCount ← heap->currentElementCount + 1
        i ← heap.currentElementCount

        // Step-2) 부모 노드와 키 값 비교 및 이동
        while ((i != 1) && (data.key > heap->pElement[i/2].key) ) {
                heap->pElement[ i ] ← heap->pElement[i/2]
                i ← i /2
        }
        heap->pElement[ i ] ← data
}
```

5. 히프 (9/9)

- 히프의 구현 (계속)

```
deleteMaxHeap ( heap ) {
        // Step-1: 루트 노드의 삭제
        result ← heap->pElement[ 1 ]

        //Step-2) 트리 마지막 자리 노드의 임시 이동
        i ← heap->currentElementCount
        temp ← heap->pElement[ i ]
        heap->currentElementCount ← heap->currentElementCount - 1

        //Step-3) 자식 노드와 키 값 비교 및 이동
        parent ← 1
        child ← 2
        while(child <= heap->currentElementCount) {
                if ( (child < heap->currentElementCount)
                        && heap->pElement[ child ].key < heap->pElement[ child+1 ].key) {
                        child ←  child+1;
                }
                if (temp->key >= heap->pElement[ child ].key) {
                        break;
                }
                heap->pElement[ parent ] ← heap->pElement[ child ]
                parent ← child
                child ← child * 2
        }
        heap->pElement[ parent ] ← temp
        return result
}
```

6. 이진 탐색 트리 (1/12)

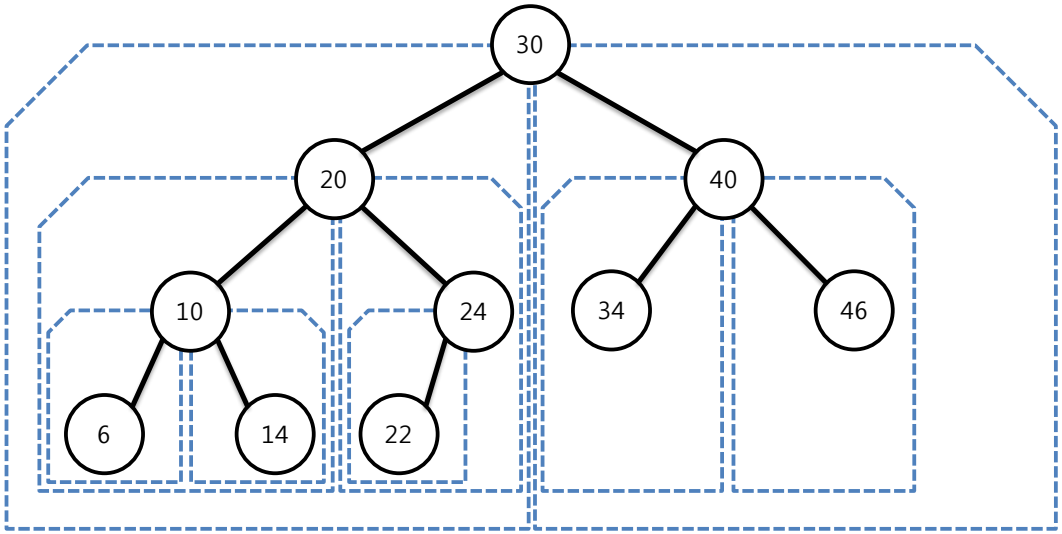
- 이진 탐색 트리
  - 목적
    - 자료의 검색
  - 특성

    1) 유일한 키 값

    2) 루트 노드의 키 값 기준

       왼쪽 서브트리 키 값

       오른쪽 서브트리 키 값



6. 이진 탐색 트리 (2/12)

- 추상 자료형
  - 이진 탐색 트리 생성
  - 이진 탐색 트리 삭제
  - 자료 추가
  - 자료 제거
  - 검색

6. 이진 탐색 트리 (4/12)

- 이진 탐색 트리의 구현 (계속)
  - 이진 탐색 트리 생성
  - 검색
  - 자료 추가
  - 자료 제거

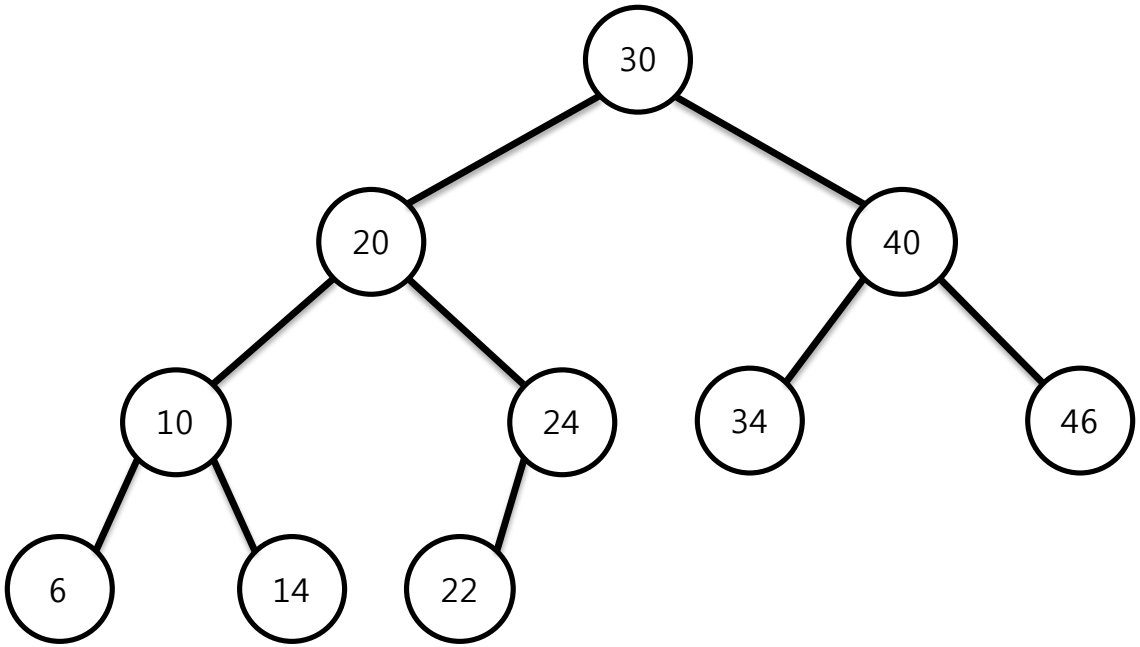
6. 이진 탐색 트리 (5/12)

- 검색 연산

```
search( tree, key ) {
        result ← tree->pRootNode
        while(result != NULL) {
                if (key == result ->key) {
                        break
                }
                else if (key < result ->key) {
                        result = result ->pLeftChild
                }
                else {
                        result = result ->pRightChild
                }
        }
        return result
}
```

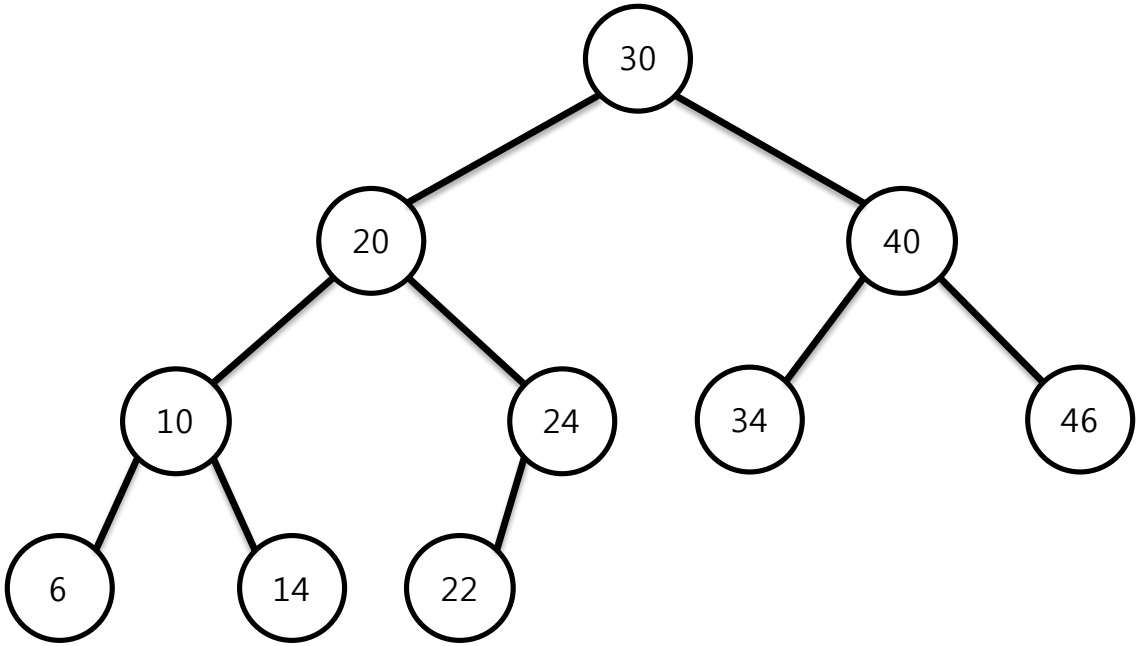
6. 이진 탐색 트리 (6/12)

- 검색의 예
  - 키 값이 22인 노드 검색
  - 키 값이 23인 노드 검색



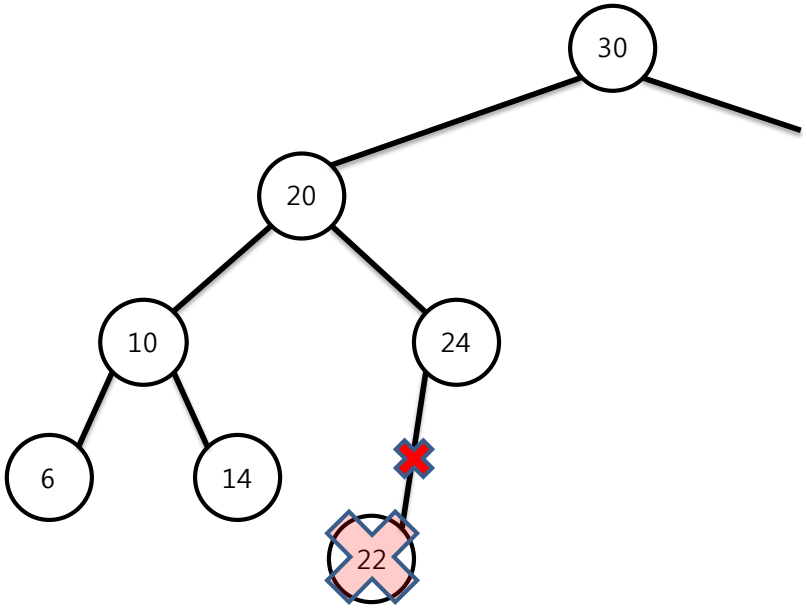
6. 이진 탐색 트리 (7/12)

- 추가 연산
  - 자료 추가를 위한 위치 찾기 (검색 연산)
    - 검색 연산 성공 → 자료 추가 불가능
    - 검색 연산 실패 → 자료 추가 가능
- ‘추가 연산’의 예
  - 키 값이 21, 23인 노드 추가



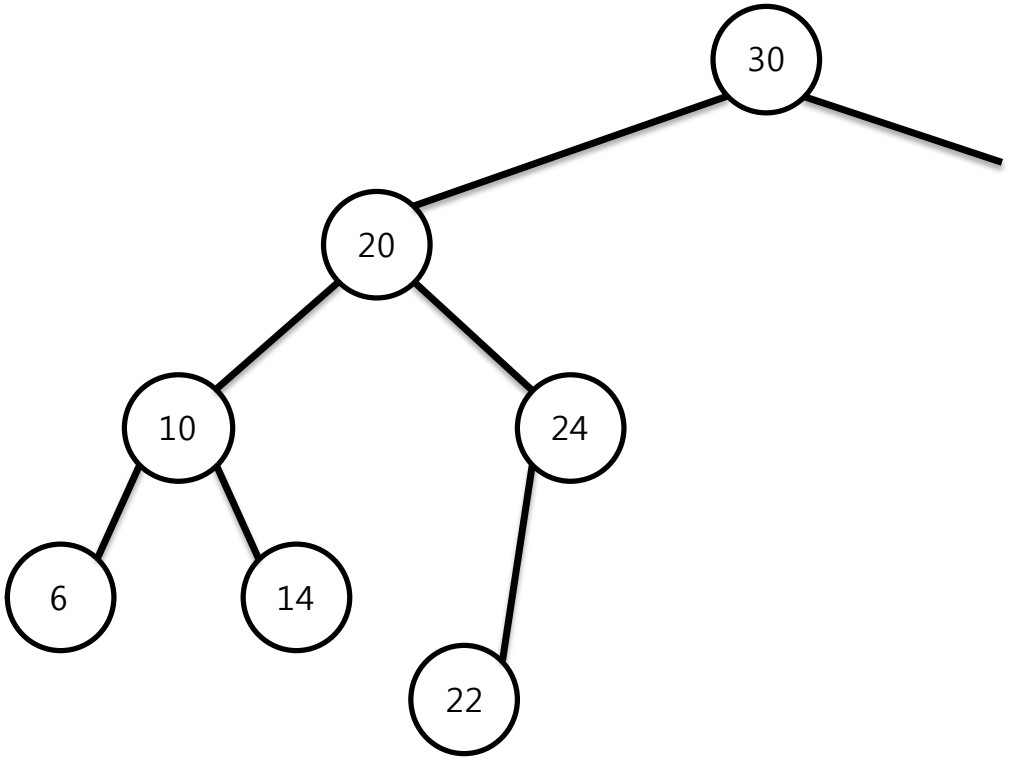
6. 이진 탐색 트리 (8/12)

- 삭제 연산, 검색 연산 먼저 실시
  - 삭제하려는 노드의 자식 노드가 없는 경우
  - 삭제하려는 노드의 자식 노드가 1개인 경우
  - 삭제하려는 노드의 자식 노드가 2개인 경우
- Case-1) 자식 노드가 없는 경우



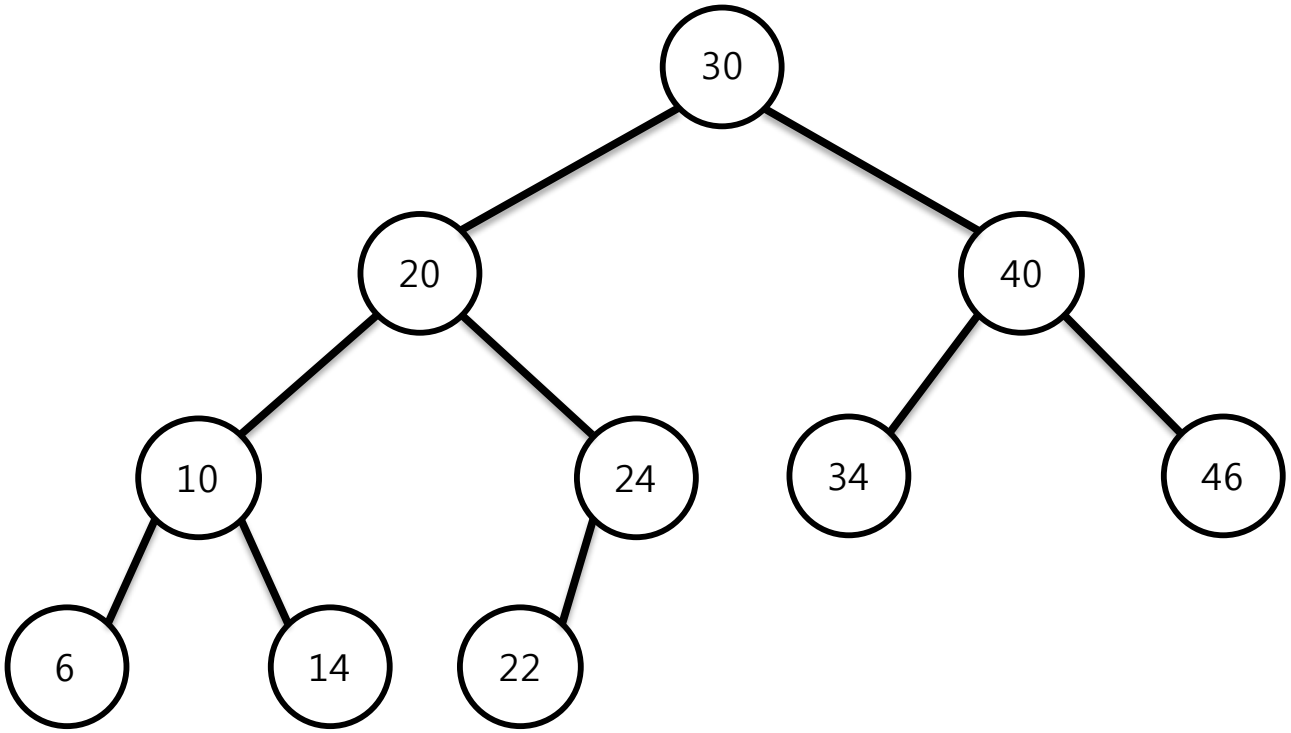
6. 이진 탐색 트리 (9/12)

- Case-2) 자식 노드가 1개인 경우



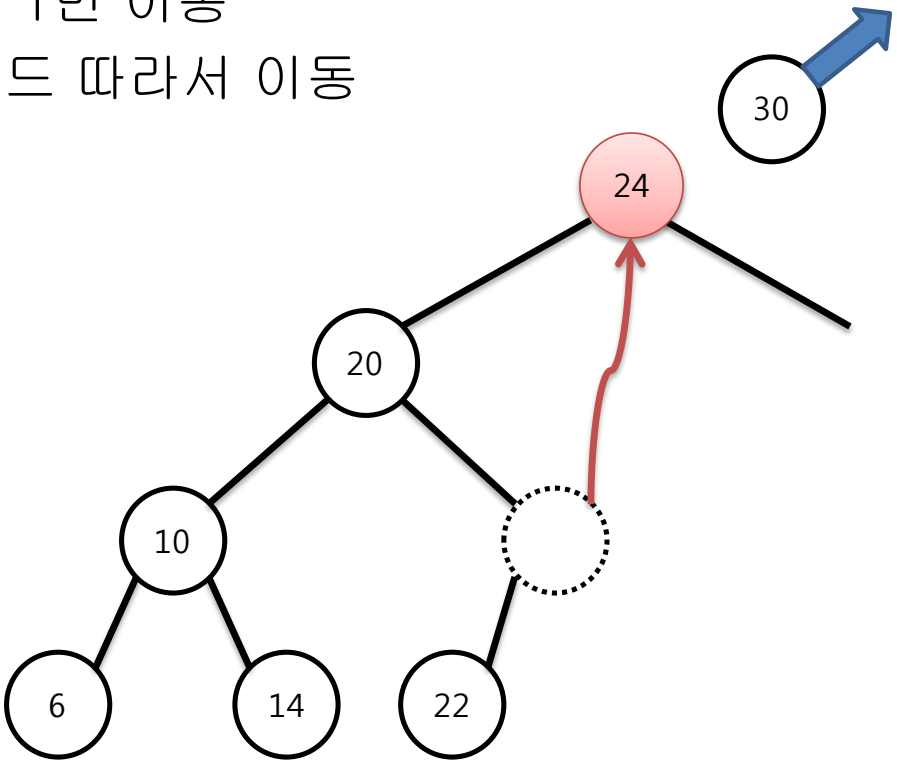
6. 이진 탐색 트리 (10/12)

- Case-3) 자식 노드가 2개인 경우
  - 왼쪽 서브트리의 가장 큰 값
  - 오른쪽 서브트리의 가장 작은 값



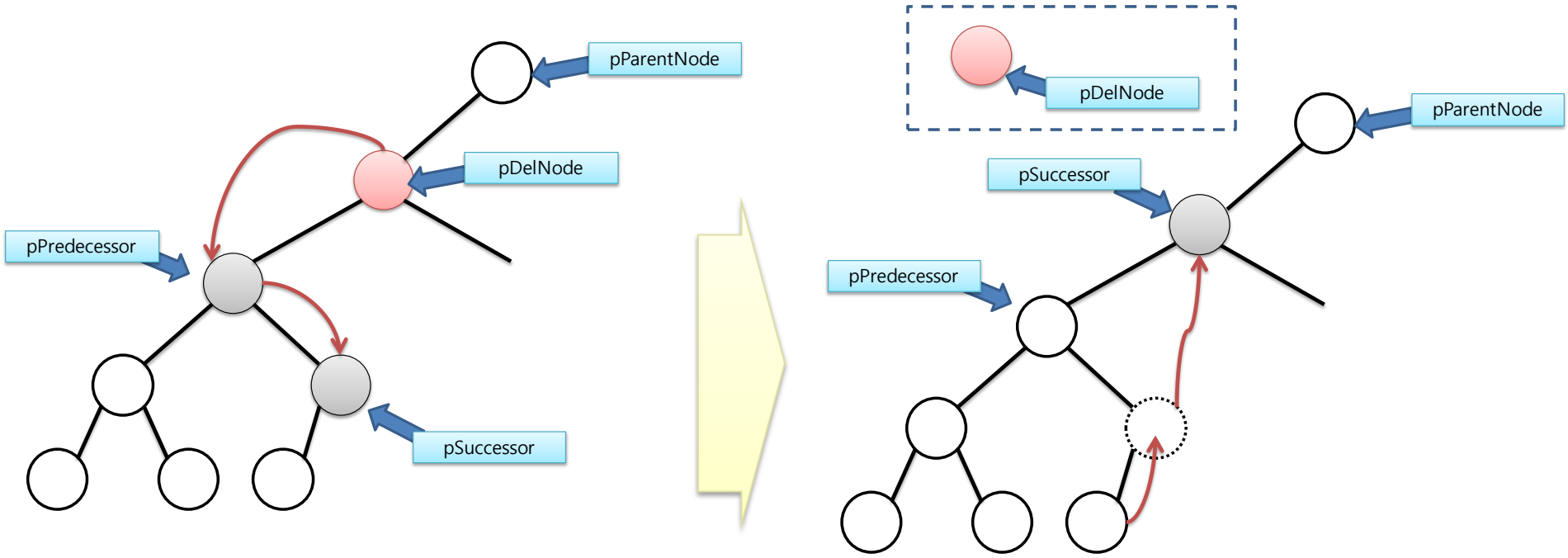
6. 이진 탐색 트리 (11/12)

- Case-3) 자식 노드가 2개인 경우 (계속)
  - 대체 노드
    - 특성: 오른쪽 자식 노드가 없다
    - 찾는 방법
      - 왼쪽 자식 노드 1번 이동
      - 오른쪽 자식 노드 따라서 이동



6. 이진 탐색 트리 (12/12)

- 이진 탐색 트리의 구현 (계속)
  - 자료 제거



이번 장에서는

- 트리의 개념
- 이진 트리
- 이진 트리의 순회
- 이진 트리 연산
- 히프
- 이진 탐색 트리