특강 2일차 - 비선형 자료구조

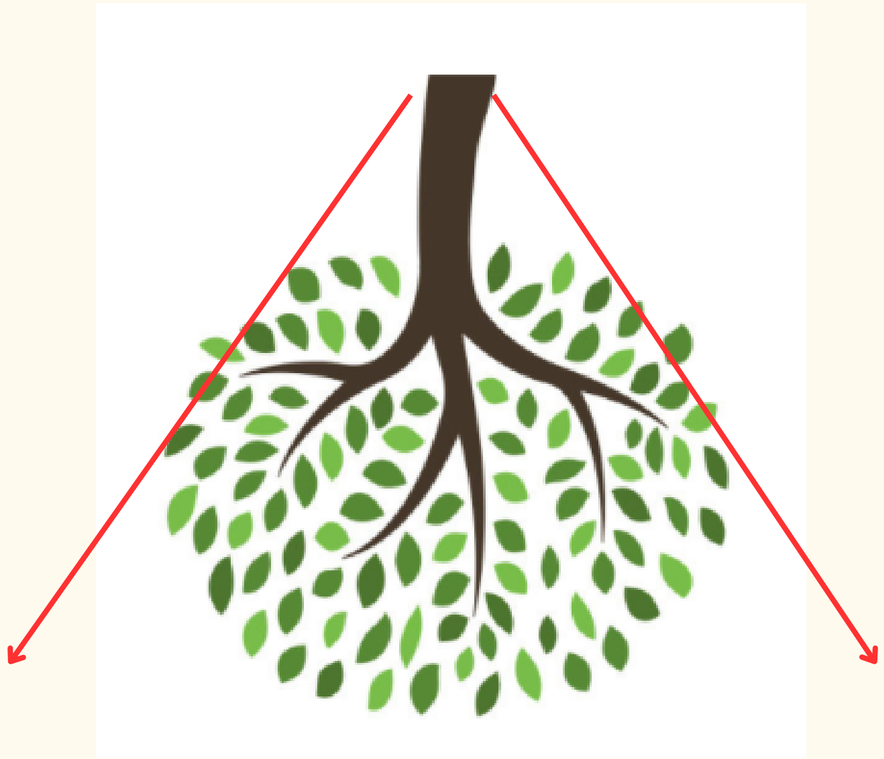
김태희

Contents

비선형 자료구조

비선형 자료구조

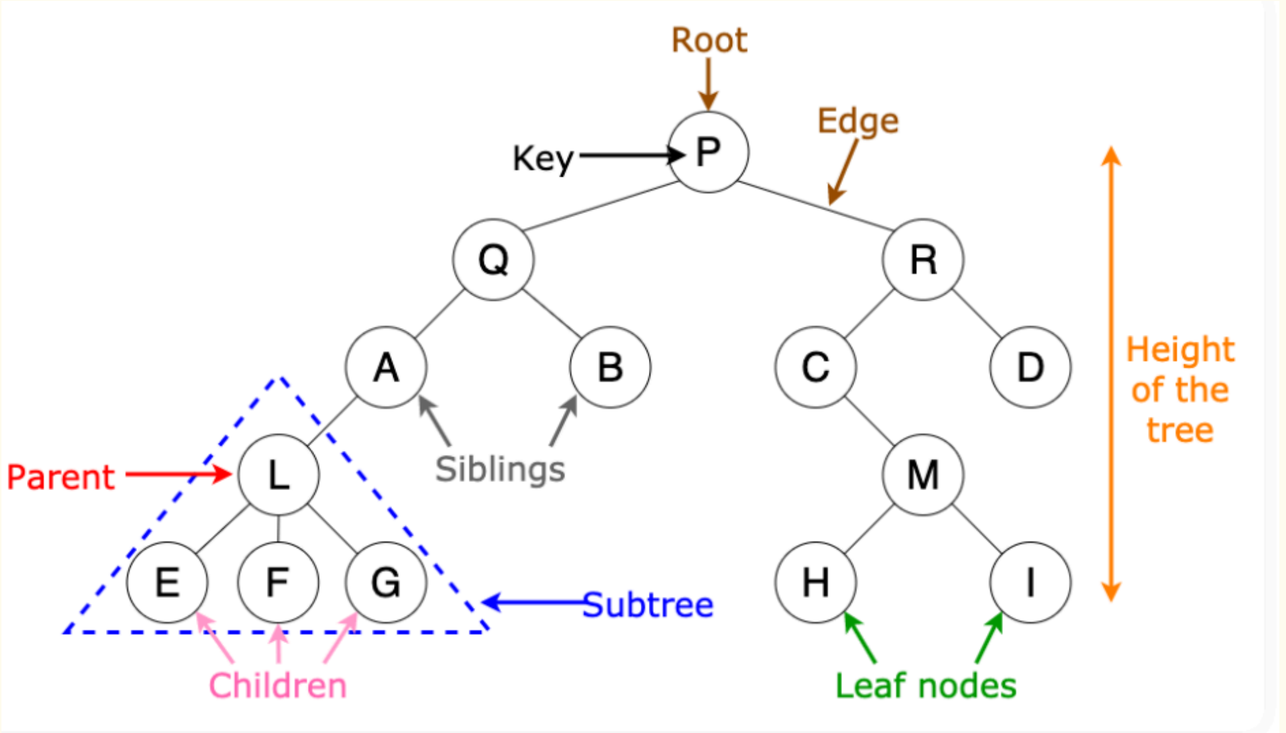
- 데이터가 순차적으로 배열되지 않고 계층적(hierarchical) 구조나 그물망(network) 구조로 구성된 자료구조
- 나무를 거꾸로 뒤집어 놓은 모양과 유사
- 특징
  - 계층적 구조: 상위와 하위 노드 간의 1:N관계로 연결
    - 트리(Tree)
  - 그물망 구조: 여러 노드가 서로 N:M관계로 연결
    - 그래프(Graph)



트리(Tree)

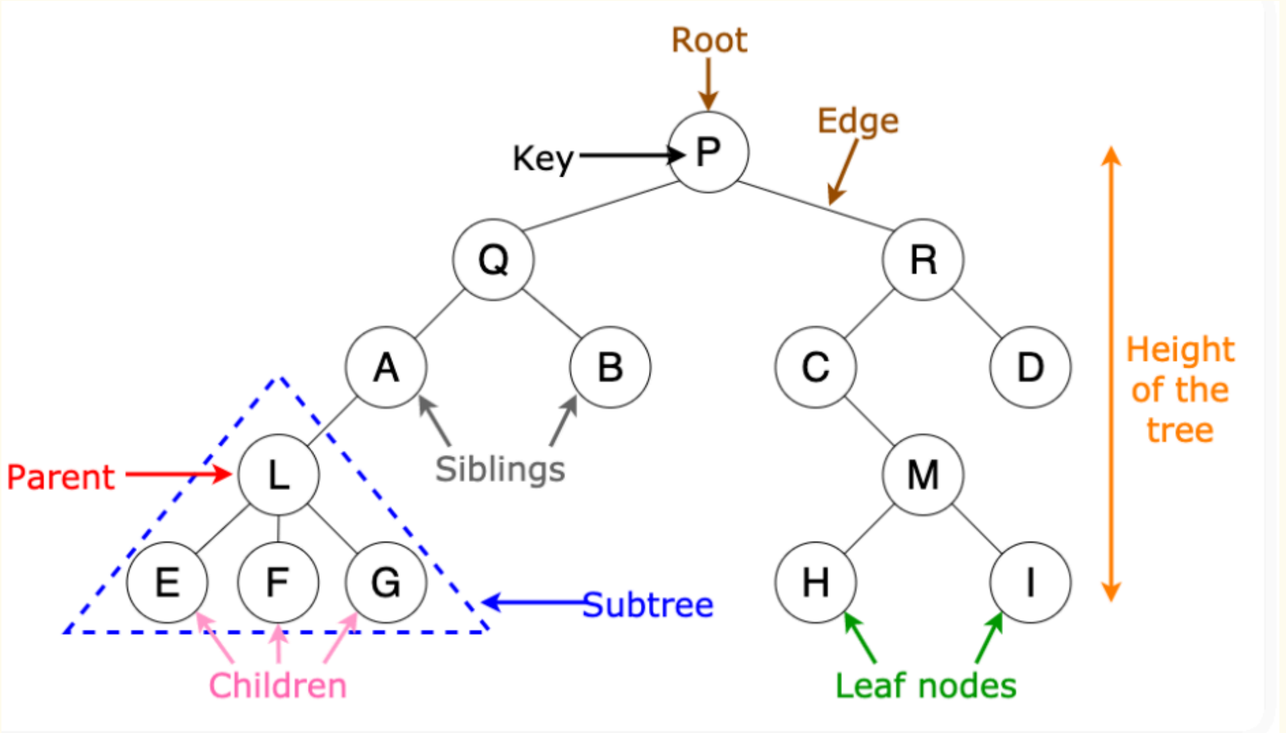
트리 용어

- 트리(Tree)
  - 1개 이상의 노드로 이루어진 유한 집합
- 노드(Node)
  - 트리의 구성 요소(단위)
- 루트 노드(Root Node)
  - 트리의 최상위 노드로, 트리의 시작점
  - 모든 다른 노드는 루트로부터 출발
  - 트리에서 유일하게 부모 노드가 없는 노드
- 간선(Edge)
  - 부모노드와 자식노드를 연결하는 선



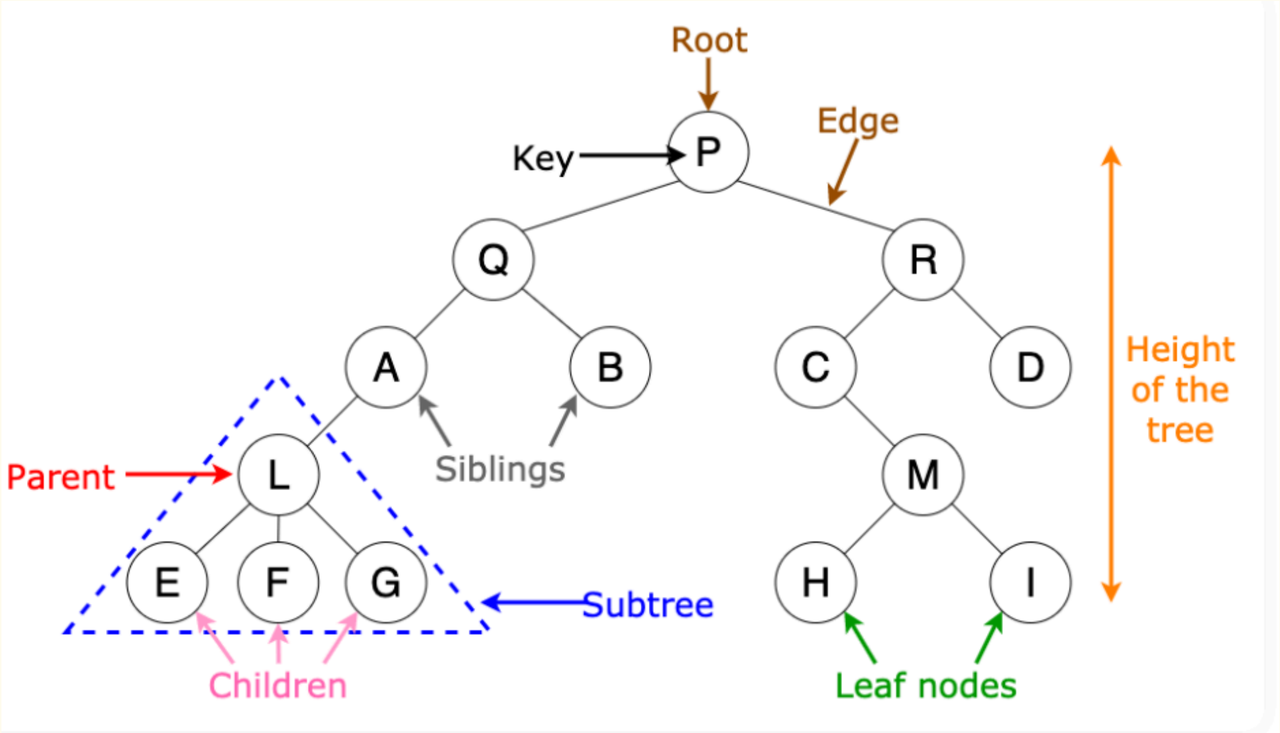
트리 용어

- 부모 노드 (Parent Node)
  - 자식 노드를 가진 노드
- 자식 노드 (Child node)
  - 부모 노드의 하위 노드
- 형제 노드 (Sibling node)
  - 같은 부모를 가지는 노드
- 단말 노드 (Terminal node), 리프 노드(Leaf node)
  - 자식 노드가 없는 노드
- 부 트리(Sub Tree)
  - 부모 노드와 연결된 간선을 끊었을 때 생성되는 트리



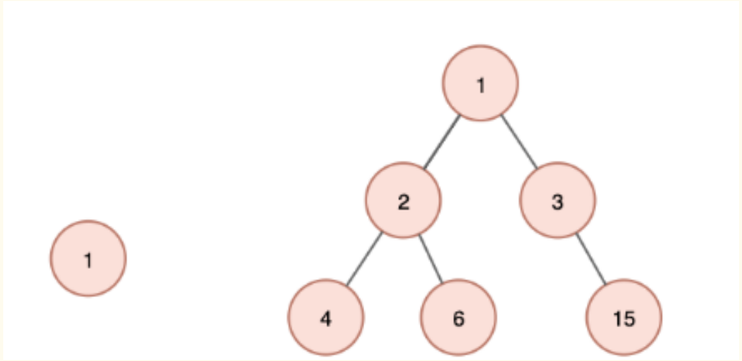
트리 용어

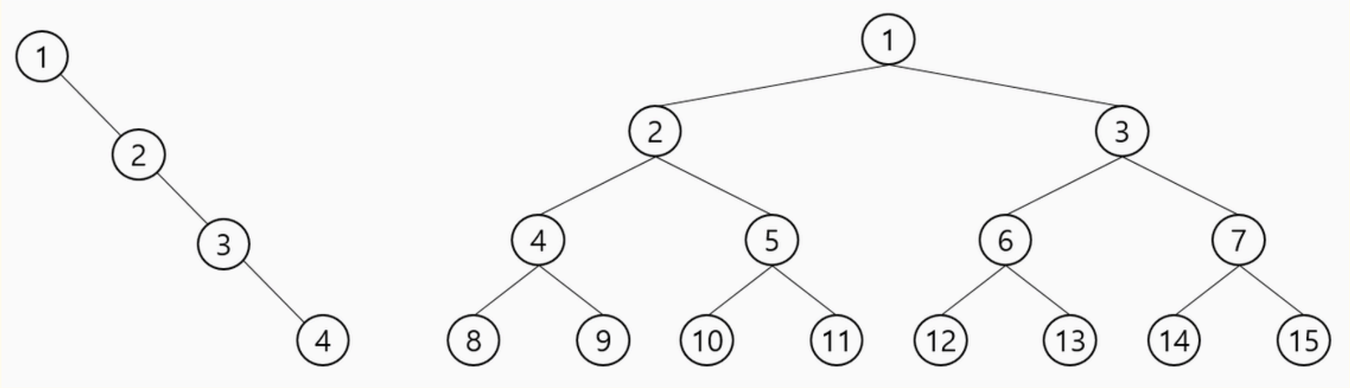
- 차수 (Degree)
  - 노드의 차수 : 노드에 연결된 자식 노드의 수
  - 트리의 차수 : MAX (노드 차수들)
- 높이 (Height)
  - 노드의 높이 : 루트에서 노드에 이르는 간선의 수
  - 트리의 높이 : MAX (노드 높이들)



이진 트리

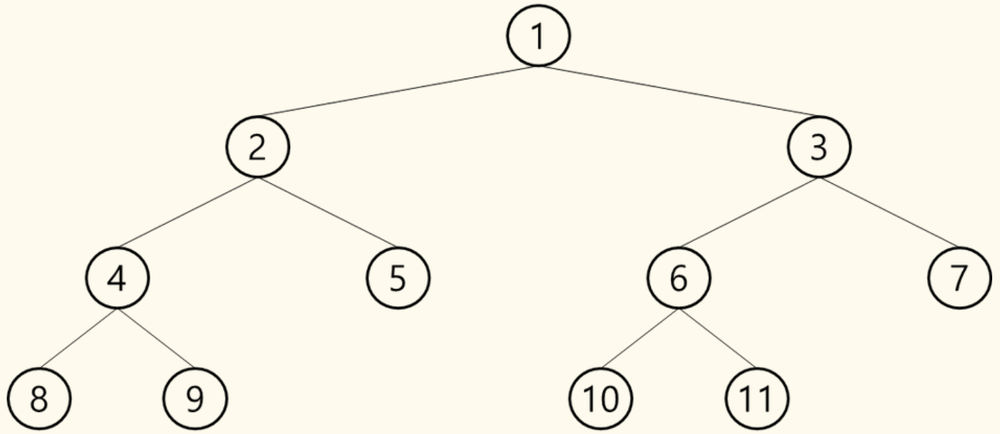
- 차수가 2인 트리
- 각 노드가 자식 노드를 최대한 2개 까지만 가질 수 있는 트리
    - 왼쪽 자식 노드(left child node) ,오른쪽 자식 노드(right child node)
- 모든 노드들이 최대 2개의 서브트리를 갖는 특별한 형태의 트리
- 높이 i에서의 노드의 최대 개수는 2^i 개
- 높이가 h인 이진 트리가 가질 수 있는 노드의 최소 개수는 (h+1)개가 되며, 최대 개수는 (2^(h+1)-1)개가 된다.



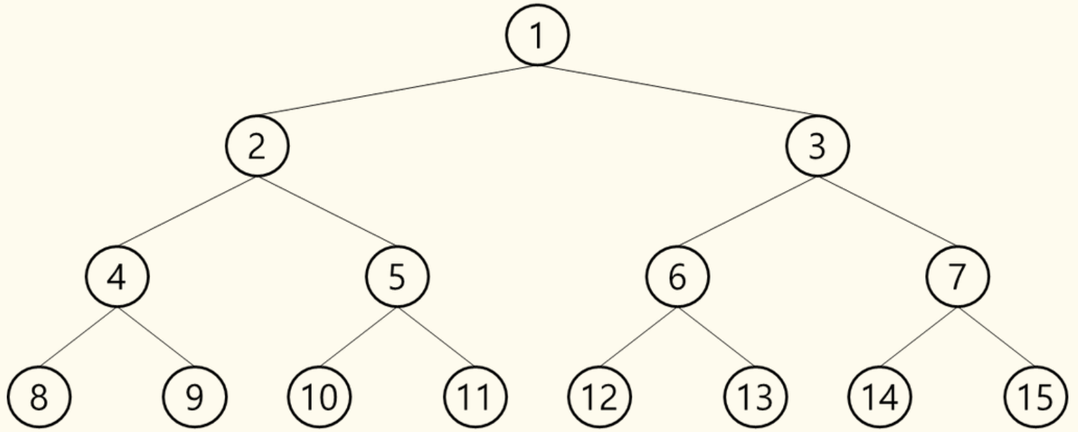


이진 트리의 종류

- 정 이진 트리(Full Binary Tree)
  - 모든 노드가 0개 또는 2개의 자식 노드를 갖는 트리

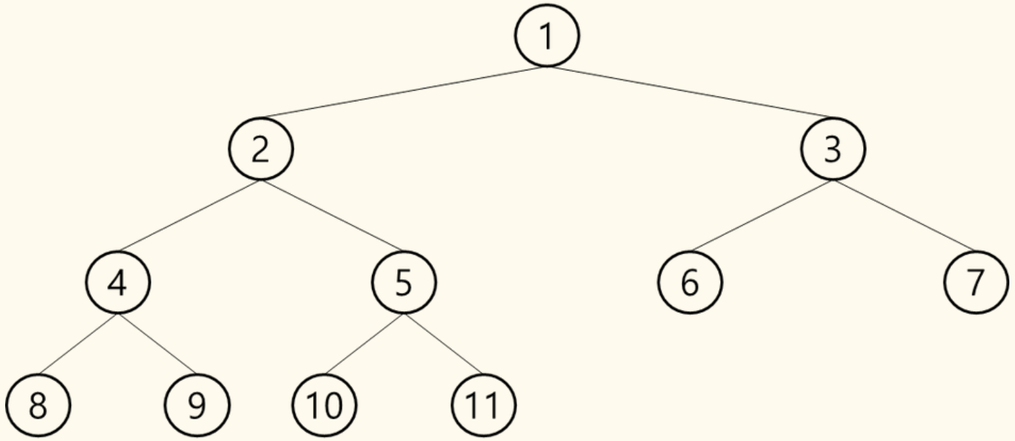


- 포화 이진 트리(Perfect Binary Tree)
  - 모든 레벨에 노드가 포화 상태로 차 있는 이진 트리
  - 높이가 h일 때, 최대의 노드 개수인 (2^(h+1)-1) 의 노드를 가진 이진 트리



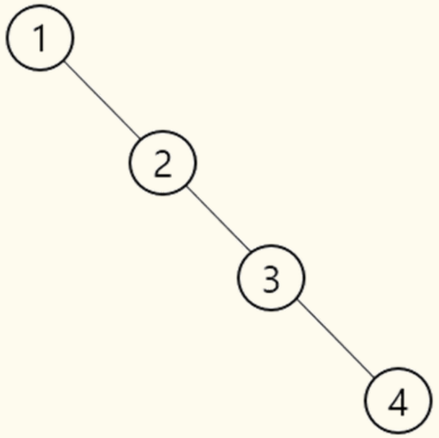
이진 트리의 종류

- 완전 이진 트리(Full Binary Tree)
  - 마지막 레벨을 제외하고 모든 레벨이 완전히 채워져 있는 트리
  - 마지막 레벨에서는 노드들이 왼쪽부터 차례로 채워진 이진 트리
  - 왼쪽부터 채워나가기 때문에 전체적으로 균형 잡힌 구조를 가짐



- 편향 이진 트리(Skewed Binary Tree)
  - 높이 h에 대한 최소 개수의 노드를 가지면서

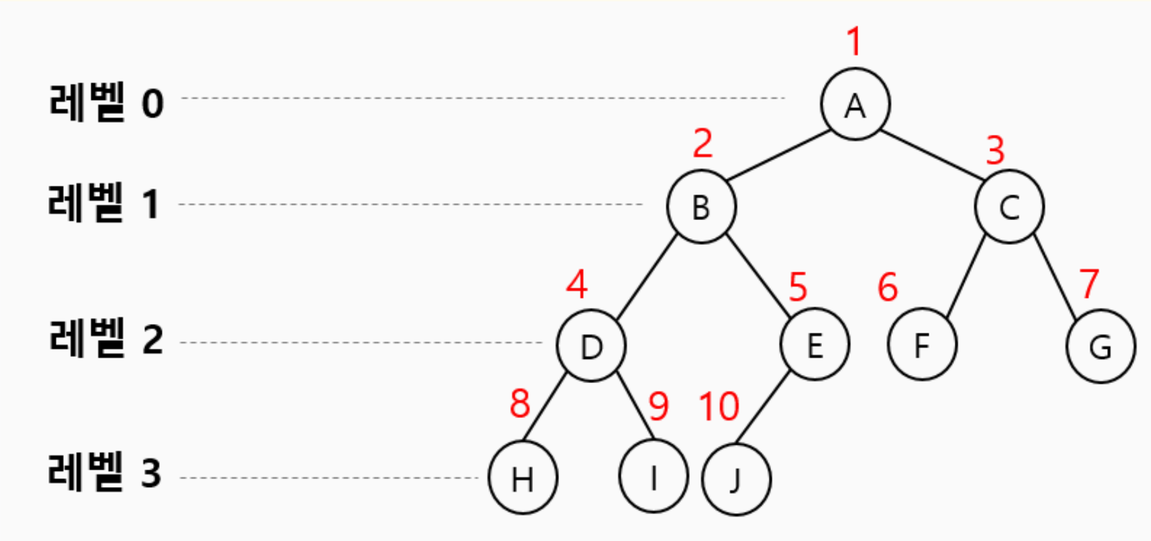
    한쪽 방향의 자식 노드 만을 가진 이진 트리



이진 트리의 표현

- 배열을 이용한 이진 트리의 표현
  - 이진 트리에 각 노드 번호를 다음과 같이 부여
  - 루트의 번호는 1로 시작
  - 레벨 n에 있는 노드에 대하여 왼쪽부터 오른쪽으로

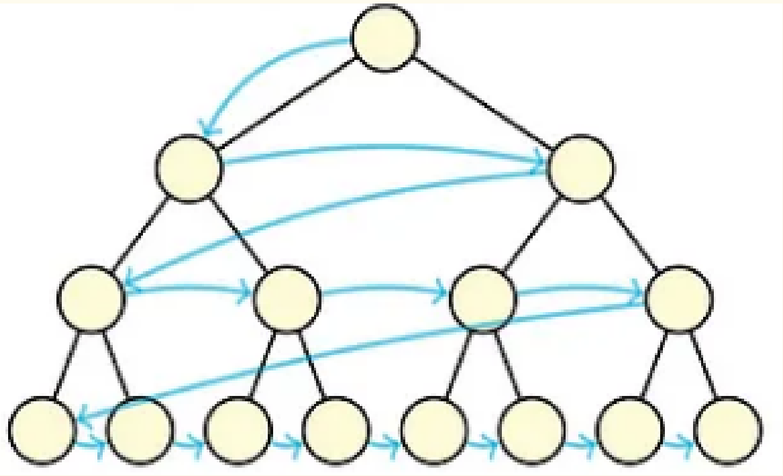
    2n 부터 2n+1 – 1 까지 번호를 차례로 부여
  - 노드 번호가 i 인 노드의 부모 노드 번호? [i / 2]
  - 노드 번호가 i 인 노드의 왼쪽 자식 노드 번호? 2 * i
  - 노드 번호가 i 인 노드의 오른쪽 자식 노드 번호? 2 * i +1



트리 탐색 - BFS

트리 탐색 - BFS

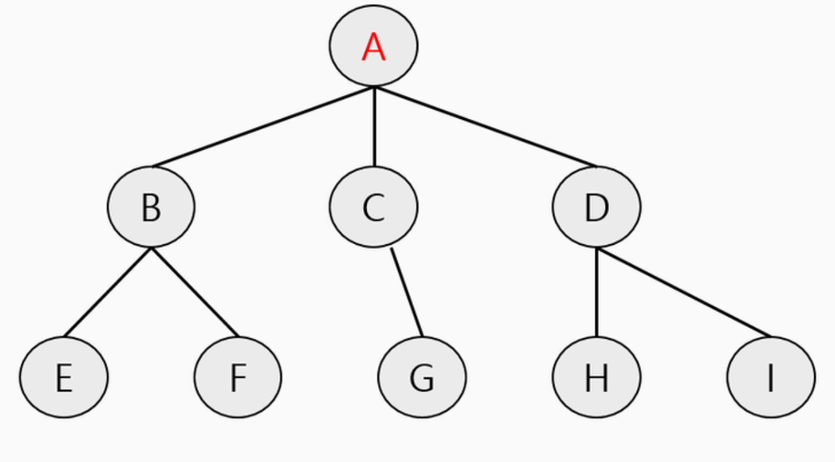
- BFS(Breadth First Search)
- 루트 노드에서 시작해 인접한 모든 자식 노드를 탐색한 후, 차례대로 다음 레벨의 노드들을 탐색하는 방식
- BFS는 레벨 우선 탐색이라고도 불리며, 트리의 모든 노드를 동일한 레벨에서 탐색한 후에 다음 레벨로 넘어감
- 큐(Queue) 자료구조를 이용해 구현



트리 탐색 - BFS

- 탐색 과정
  - 시작 노드를 큐에 삽입
  - 큐에서 노드를 꺼내어 해당 노드의 인접한 노드들을 모두 큐에 삽입
  - 큐가 빌 때까지 이 과정을 반복

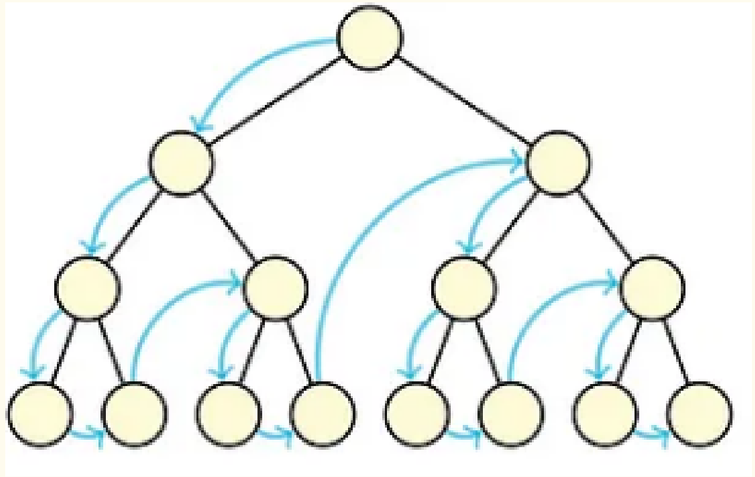
Q



트리 탐색 - DFS

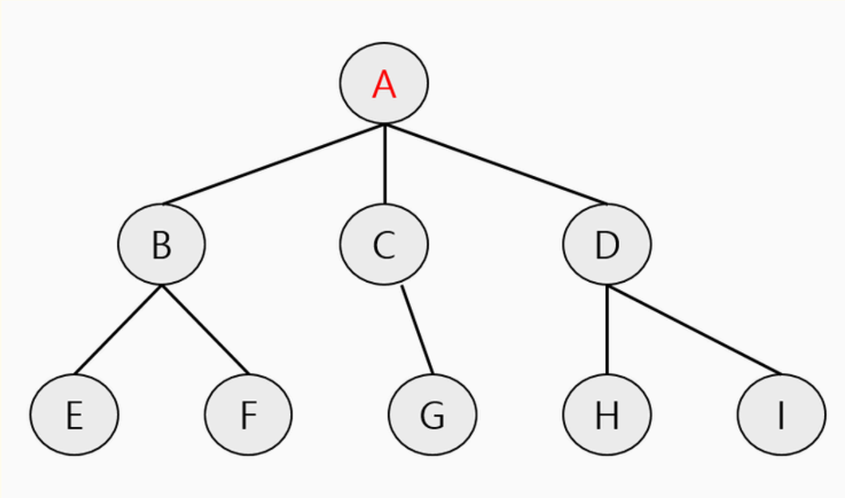
트리 탐색 - DFS

- DFS(Depth First Search)
- 한 노드에서 시작해 갈 수 있는 만큼 깊이 탐색한 후, 더 이상 갈 곳이 없으면 되돌아오는 방식
- 가장 마지막에 만났던 갈림길의 노드로 되돌아가서 다시 깊이 우선 탐색을 반복해야 함
    - 재귀적으로 구현
    - 후입선출 구조의 스택 사용해서 구현



트리 탐색 - DFS

- 탐색 과정
  - 루트 노드부터 탐색 시작
  - 현재 노드에서 갈 수 있는 인접한 노드 중 하나를 선택하여 탐색
  - 더 이상 갈 수 있는 노드가 없으면, 이전 노드로 되돌아가서 다른 노드를 탐색
  - 모든 노드를 방문할 때까지 이 과정을 반복

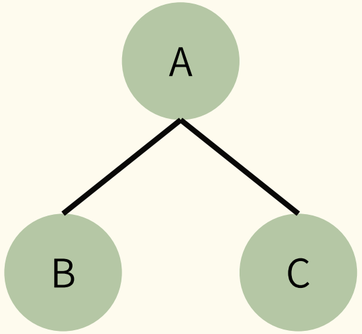



```
탐색(v)
    v 방문;
    for(v의 모든 자식노드 w) {
        탐색(w);
    }
```

이진 트리 탐색

이진 트리 탐색

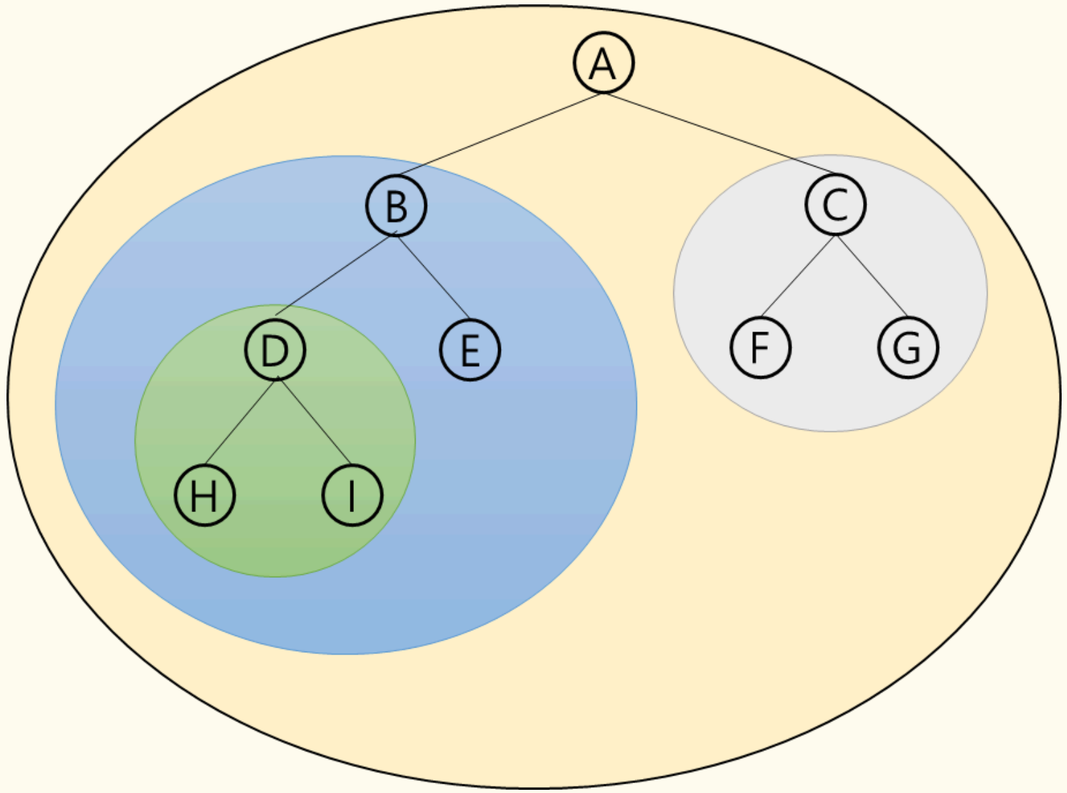
- 전위(Preorder) 순회
  - **Root**-Left-Right
  - 부모노드 방문 후, 자식노드를 좌,우 순서로 방문
- 중위(InOrder) 순회
  - Left-**Root**-Right
  - 왼쪽 자식노드 , 부모노드, 오른쪽 자식노드 순으로 방문
- 후위(PostOrder) 순회
  - Left-Right-**Root**
  - 왼쪽 자식노드 , 오른쪽 자식노드, 부모 노드 순으로 방문



전위 순회 : A - B - C
중위 순회 : B - A - C
후위 순회 : B - C - A

이진 트리 탐색

- 다음 그래프의 전위, 중위, 후위 순회 결과는?



이진 트리의 활용

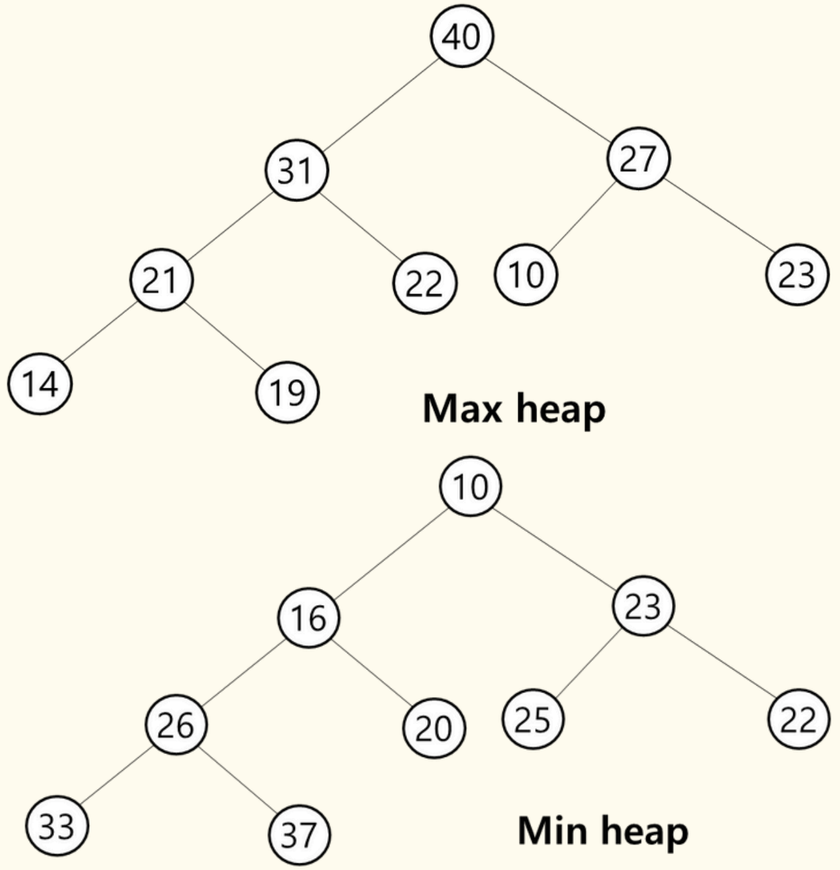
- 수식 트리
- 이진 탐색 트리

- 활용
  - 부동산 다툼
    - https://www.acmicpc.net/problem/19638

힙(Heap)

힙(Heap)

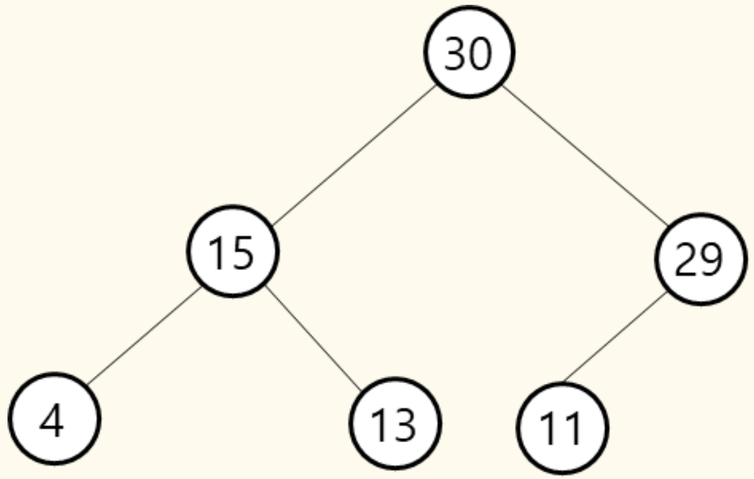
- 완전 이진 트리에 있는 노드 중에서 키 값이 가장 큰 노드나 키 값이 가장 작은 노드를 찾기 위해서 만든 자료구조
- 최대 힙(max heap)
  - 키 값이 가장 큰 노드를 찾기 위한 완전 이진 트리
  - 부모 노드의 키 값 >= 자식 노드의 키 값
  - 루트 노드 : 키 값이 가장 큰 노드
- 최소 힙(min heap)
  - 키 값이 가장 작은 노드를 찾기 위한 완전 이진 트리
  - 부모 노드의 키 값 =< 자식 노드의 키 값
  - 루트 노드 : 키 값이 가장 작은 노드



힙(Heap) 연산

- 원소 추가 : O(logN)
  - 20을 추가한다면?
  - 50을 추가한다면?

- 원소 삭제 : O(logN)



힙(Heap)의 활용

- 힙 정렬
- 우선 순위 큐(Priority Queue)
  - java.util.PriorityQueue
- 활용
  - 센티와 마법의 뽕망치
    - https://www.acmicpc.net/problem/19638

Thank you!