파이썬으로 배우는 데이터 구조

한동대학교
전산전자공학부
김영섭 교수

학습 목표

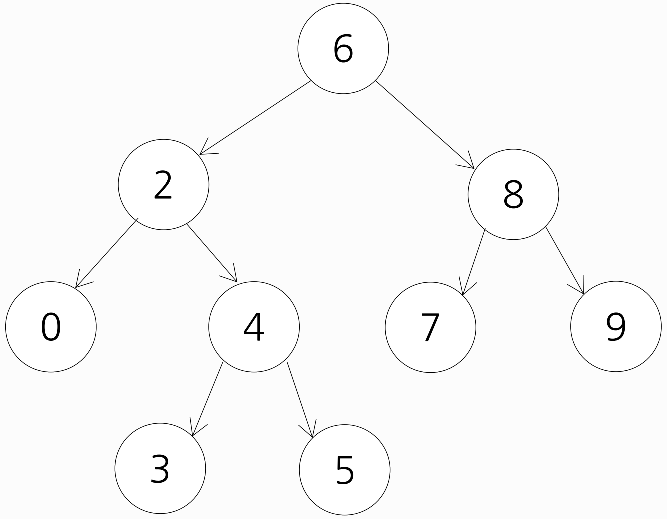
BST의 다양한 메소드들을 이해하고 구현할 수 있다

Data Structures in Python
Chapter 7 - 2

- Binary Search Tree(BST)
- **BST Algorithms**
- AVL Tree
- AVL Algorithms

LCA

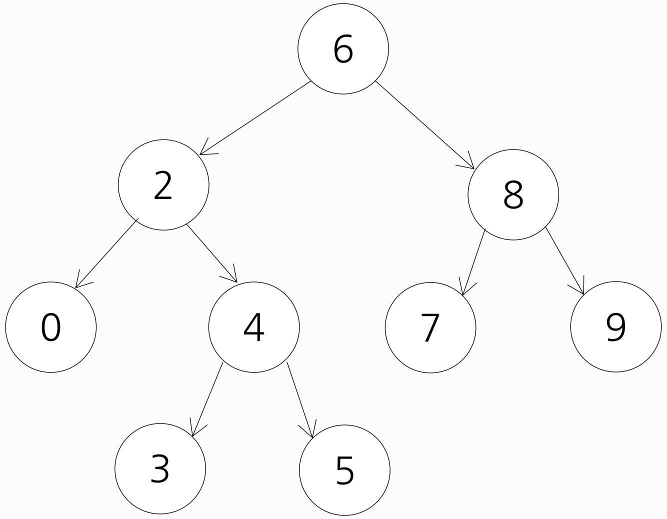
- Find the lowest common ancestor(LCA) of two given nodes, given in BST.
  - The LCA is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow a node to be a descendant of itself).”
  - In BST, all of the nodes' values will be unique.
    Two nodes given, p and q, are different and both values will exist in the BST.



```
For example:
2, 8 -> 6
2, 5 -> 2
9, 5 -> 6
8, 7 -> 8
0, 5 -> 2
```

LCA - iteration

- **Intuition (Iteration)**: Traverse down the tree iteratively to **find the split point**. The point from where p and q won't be part of the same subtree or when one is the parent of the other.

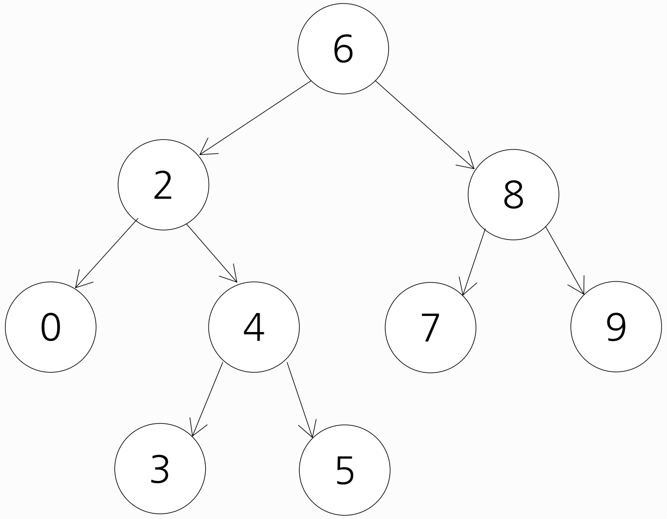


```
For example:
2, 5 -> 2
9, 7 -> 8
0, 4 -> 2
0, 5 -> 2
2, 7 -> 6
```

```
def LCAiteration(self, p, q):
    while node != None:
        if both p & q > node:
            node move to right to search
        elif both q & q < node
            node moves to left to search
        else
            return node.key    # found
    return None                # not found
```

LCA - recursion

- **Algorithm: (Recursion)**
  1. Start traversing the tree from the root node.
  2. If both the nodes p and q are in the right subtree, then continue the search with right subtree starting step 1.
  3. If both the nodes p and q are in the left subtree, then continue the search with left subtree starting step 1.
  4. If both step 2 and step 3 are **not true,** this means we have **found** the node which is common to node p's and q's subtrees. Hence we return this common node as the LCA.



```
def LCA(self, p, q):


  # your code here
```

LCA complexity

- Recursion Algorithm
  - Time Complexity: $O(N)$, where N is the number of nodes in the BST.
    In the worst case we might be visiting all the nodes of the BST.
  - Space Complexity: $O(N)$. This is because the maximum amount of space utilized by the recursion stack would be N since the height of a skewed BST could be N.
- Iteration Algorithm
  - Time Complexity : O(N), where N is the number of nodes in the BST.
    In the worst case we might be visiting all the nodes of the BST.
  - Space Complexity : O(1).

학습 정리

1) Predecessor, successor는 트리의 root를 삭제할 경우, 대체할 값을 찾기 위해 사용된다

2) 노드를 삭제할 때는 no child, one child, two children의 세가지 경우로 나누어 생각한다

파이썬으로 배우는 데이터 구조