# 2020년 가을학기 알고리즘

## Lowest Common Ancestor

데이터네트워크연구실
문현수, 이영석
munhyunsu@cs-cnu.org

# 9주차 Feedback

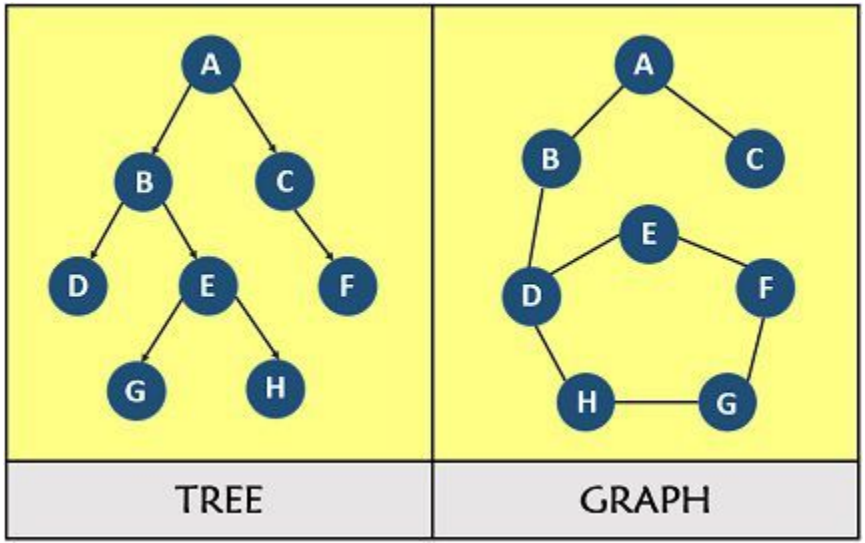
- 제출률: 82%
- 구글 코딩테스트 (중요한 에지) 어려움
  - 기말고사 전 코드 리뷰 시간에 다룰 예정

# 이번주 실습 목표

- 최소 공통 조상 노드
  - Tree
  - Tree Traversal
  - Lowest Common Ancestor

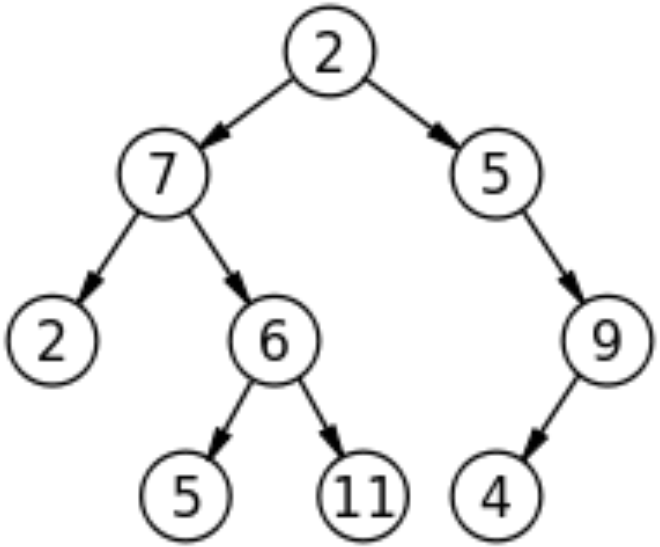
- 실습1: Find Root (1점)
- 실습2: Tree Traversal (2점)
- 실습3: Lowest Common Ancestor (3점)
- 실습4: Deepest Subtree (4점)

# Graph vs. Tree (Directed Acyclic Graphs)



# Binary Tree

- 최대 2개의 '자식' 노드를 가지는 그래프 (트리)




# 코딩테스트에서는 불리하지만, 확장성있는 구현법

```
public class BinaryTree {$
    Map<String, Node> nodes;$
    Node root;$
$
    BinaryTree() {$
        this.nodes = new HashMap<>();$
        this.root = null;$
    }$
```

```
class BinaryTree(object):$
    def __init__(self):$
        self.nodes = dict()$
        self.root = None$
```

```
public class Node {$
    String key;$
    Node left;$
    Node right;$
$
    Node(String key) {$
        this.key = key;$
        this.left = null;$
        this.right = null;$
    }$
```

```
class Node(object):$
    def __init__(self, key, left=None, right=None):$
        self.key = key$
        self.left = left$
        self.right = right$
```

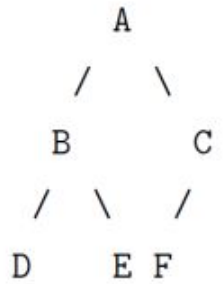
# 실습1. Find Root (1)

# 루트 찾기 - 트리 생성 연습

- 이진 트리의 노드의 개수가 주어지고, 노드의 key와 자식들이 주어졌을 때 해당 트리의 꼭대기 (Root) 를 찾으시오.
    - 노드의 순서는 무작위로 입력된다.

[Input] (오른쪽 사진: 이론)

```
6
D . .
E . .
F . .
C F .
B D E
A B C
```





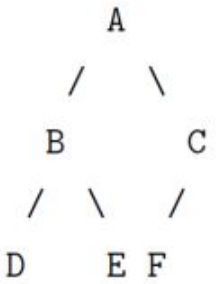
[Output]

```
A
```

# 실습2. Tree Traversal (2)

# 트리 순회 - 트리 다루기 연습

- 이진 트리의 노드의 개수가 주어지고, 노드의 key와 자식들이 주어졌을 때 해당 트리를 Preorder / Inorder / Postorder 순으로 순회하고 출력하시오.





[Input] (오른쪽 사진: 이론)
6
D . .
E . .
F . .
C F .
B D E
A B C

[Output]
A B D E C F
D B E A F C
D E B F C A

# Tree Traversal (WiKi)

```
preorder(node)
    if (node == null)
        return
    visit(node)
    preorder(node.left)
    preorder(node.right)
```

```
iterativePreorder(node)
  if (node == null)
    return
  s ← empty stack
  s.push(node)
  while (not s.isEmpty())
    node ← s.pop()
    visit(node)
    //right child is pushed first so that left is processed first
    if node.right ≠ null
      s.push(node.right)
    if node.left ≠ null
      s.push(node.left)
```

```
inorder(node)
    if (node == null)
        return
    inorder(node.left)
    visit(node)
    inorder(node.right)
```

```
iterativeInorder(node)
  s ← empty stack
  while (not s.isEmpty() or node ≠ null)
    if (node ≠ null)
      s.push(node)
      node ← node.left
    else
      node ← s.pop()
      visit(node)
      node ← node.right
```

```
postorder(node)
    if (node == null)
        return
    postorder(node.left)
    postorder(node.right)
    visit(node)
```

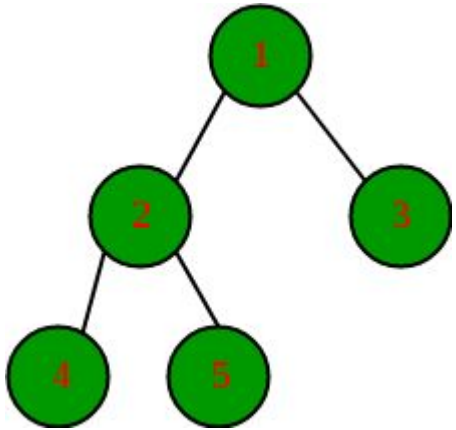
```
iterativePostorder(node)
  s ← empty stack
  lastNodeVisited ← null
  while (not s.isEmpty() or node ≠ null)
    if (node ≠ null)
      s.push(node)
      node ← node.left
    else
      peekNode ← s.peek()
      // if right child exists and traversing node
      // from left child, then move right
      if (peekNode.right ≠ null and lastNodeVisited ≠ peekNode.right)
        node ← peekNode.right
      else
        visit(peekNode)
        lastNodeVisited ← s.pop()
```

# 실습3. Lowest Common Ancestor (3)

# 최소 공통 조상 노드 - 트리 연산

- 이진 트리의 노드의 개수가 주어지고, 노드의 key와 자식들이 주어졌을 때 해당 트리를 구축하고, 질의 노드의 최소 공통 조상 노드를 출력하시오.
    - 이론 슬라이드 참고!
    - 질의 노드 2개는 같을 수 있다.

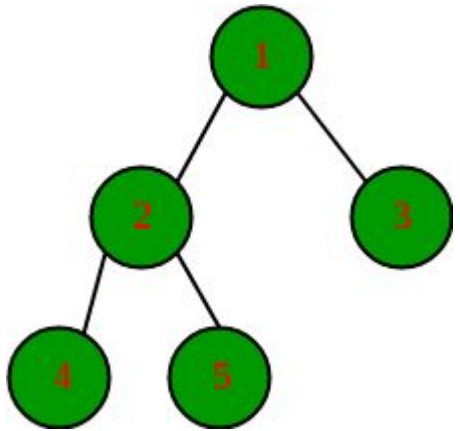
```
[Input] (오른쪽 사진: 이론)
5
4 . .
3 . .
2 4 5
1 2 3
5 . .
3 4

[Output]
1
```



# 실습4. Deepest Subtree (4)

# 서브 트리 - 트리 연산 및 활용

- 이진 트리의 노드의 개수가 주어지고, 노드의 key와 자식들이 주어졌을 때 해당 트리를 구축하고, 가장 깊은 노드를 모두 포함하는 서브 트리의 '루트'를 출력하시오.




```
[Input] (오른쪽 사진: 이론)
5
4 . .
3 . .
2 4 5
1 2 3
5 . .

[Output]
2
```

# 참고

# Standard Library

- Python 3: https://docs.python.org/3/library/
- JAVA 11: https://cr.openjdk.java.net/~iris/se/11/latestSpec/api/
- C: https://en.cppreference.com/w/c
- CPP: https://en.cppreference.com/w/cpp

| ID/ext | external ID | name |
|---|---|---|
| adb | ada | Ada |
| awk | awk | AWK |
| bash | bash | Bash shell |
| c | c | C |
| csharp | csharp | C# |
| cpp | cpp | C++ |
| f95 | f95 | Fortran |
| hs | haskell | Haskell |
| java | java | Java |
| js | javascript | JavaScript |
| kt | kotlin | Kotlin |
| lua | lua | Lua |
| pas | pascal | Pascal |
| pl | pl | Perl |
| sh | sh | POSIX shell |
| plg | prolog | Prolog |
| py2 | python2 | Python 2 |
| py3 | python3 | Python 3 |
| r | r | R |
| rb | ruby | Ruby |
| scala | scala | Scala |
| swift | swift | Swift |

# 주요 링크

- 코딩테스트: http://coding.cnu.ac.kr:8080/domjudge/public
- FAQ : https://docs.google.com/document/d/1ntR6GS1SI7dRbYlw-pu8uT8U65Wc-RtMj10IEpiapfU/
- 질문: 메일!

- 문의 사항: munhyunsu@cs-cnu.org
  - ID / PASSWORD 변경
  - 실습 시간외 질문: [AL]질문제목

# 잊지 않아야 할 것) 소스코드 및 보고서

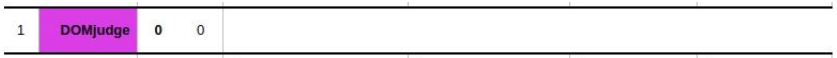
- 사이버 캠퍼스에 목요일까지 제출
  - 추가 시간 필요한 학생들도 목요일까지 제출. 추가 시간 문제 해결은 메일로도 제출!
- 보고서(.pdf 파일), 소스코드(.java, .py 등) zip 파일 압축
  - AL_학번_이름_06.zip (메일 추가 제출) or AL19_06.zip (사이버캠퍼스)
- 시간/공간 복잡도 해석(STL 고려), 자신의 생각, 질문, 느낀점, 공유하고 싶은 문제
  - 문제 해결을 위해 어떤 접근법을 사용하였는지, 무엇을 배웠고 느꼈는가?

# 어디에 제출해야하는가? 헷갈린다.

- 제한 시간 내에 모두 해결했는가?
  - 네 - 사이버캠퍼스에 소스코드와 보고서 압축해서 제출
  - 아니요 -
    - 추가 시간 내에 해결한 문제가 있는가?
      - 네 - 사이버캠퍼스에 소스코드와 보고서 압축해서 제출하고, 메일로도 제출
      - 아니요 - 사이버캠퍼스에 소스코드와 보고서 압축해서 제출

# 보고서 템플릿: .pdf 로 제출!

- 알고리즘-x주차-주제
  학번 이름
- 코드 테스팅 결과 (점수표)




- 각 문제별 내용
  a. 문제 / 목표
  b. 해결 방법 (주요 소스코드 첨부. 스크린샷 or 코드 CV)
  c. 결과 (입력, 출력 결과)
- 느낀점: 과제를 하며 느낀 점 / 공유하고싶은 문제 / 난이도 / 부탁 / 조교에게... 등