알고리즘 실습

181101 - Tree Traversal

오늘의 목표

- 트리(Tree)란 무엇인가?
  - 루트(Root), 리프(Leaf), 엣지(Edge), 부모, 자식
  - 이진트리
- 트리 순회 알고리즘
  - 너비 우선 탐색
  - 깊이 우선 탐색

Feedback

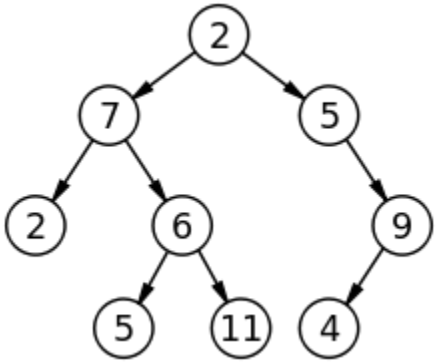
지난 시간: 중.간.고.사.

- 구체적인 피드백은 다음주에...
  - 채점 기준표
  - 점수

트리 표현

Tree

- 방향이 있게 연결된 노드들: 순환, 다중 연결 없음
  - https://en.wikipedia.org/wiki/Tree_(data_structure)



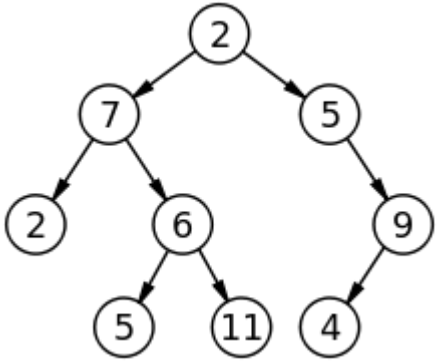
Tree 구성: 노드 클래스

- node.py
  - 생성자(초기화)
  - getter / setter
  - __str__(self) ?
    - JAVA에서의 toString() 같은 것

```python
class Node(object):
    def __init__(self, key=None, value=None, left=None, right=None):
        self._key = key
        self._value = value
        self._left = left
        self._right = right

    def set_key(self, key):
        self._key = key

    def get_key(self):
        return self._key

    def set_value(self, value):
        self._value = value

    def get_value(self):
        return self._value

    def set_left(self, left):
        self._left = left

    def get_left(self):
        return self._left

    def set_right(self, right):
        self._right = right

    def get_right(self):
        return self._right

    def __str__(self):
        return str({'key': self._key, 'value': self._value,
                    'left': self._left, 'right': self._right})
```

Tree 그리기

- 수동으로 그려보자
- value는 우선 무시




```
37  def main():
38      tree = Node(key=2)
39
40      tree.set_left(Node(key=7))
41      tree.set_right(Node(key=5))
42
43      tree.get_left().set_left(Node(key=2))
44      tree.get_left().set_right(Node(key=6))
45      tree.get_right().set_right(Node(key=9))
46
47      tree.get_left().get_right().set_left(Node(key=5))
48      tree.get_left().get_right().set_right(Node(key=11))
49      tree.get_right().get_right().set_left(Node(key=4))
50
51      print(tree)
52
53
54  if __name__ == '__main__':
55      main()
56
```

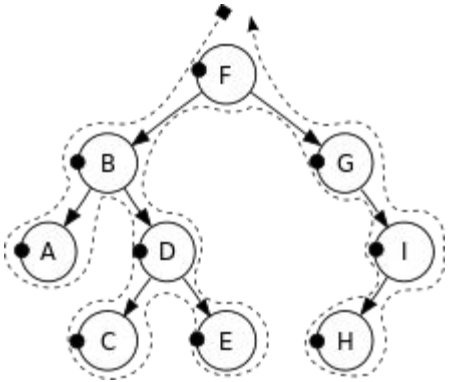
```
/home/harny/PycharmProjects/SCSC/venv/bin/python /home/harny/PycharmProjects/SCSC/week8/node.py
{'key': 2, 'value': None, 'left': <__main__.Node object at 0x7fde01b5c518>, 'right': <__main__.Node object at 0x7fde01bb9320>}
```

트리 순회

Pre Order

- 자식노드 순회 전에 처리
  - F, B, A, D, C, E, G, I, H.

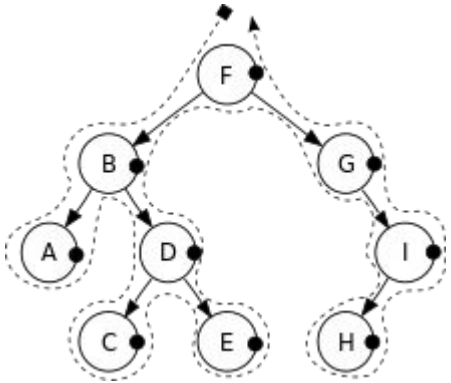
```
preorder(node)
  if (node = null)
    return
  visit(node)
  preorder(node.left)
  preorder(node.right)
```



In Order

- 자식노드 순회 중에 처리
  - A, B, C, D, E, F, G, H, I.

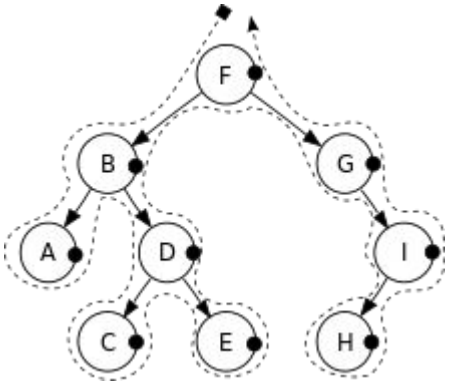
```
inorder(node)
  if (node = null)
    return
  inorder(node.left)
  visit(node)
  inorder(node.right)
```



Post Order

- 자식노드 순회 후에 처리
  - A, C, E, D, B, H, I, G, F.

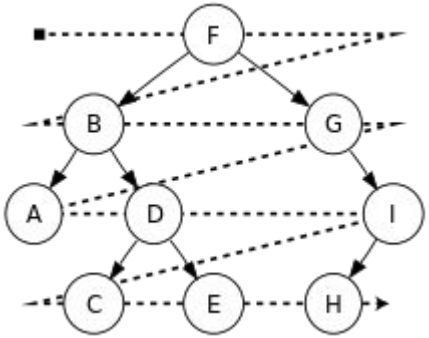
```
postorder(node)
  if (node = null)
    return
  postorder(node.left)
  postorder(node.right)
  visit(node)
```



Level Order

- 같은 depth끼리 왼쪽부터 출력
  - F, B, G, A, D, I, C, E, H.

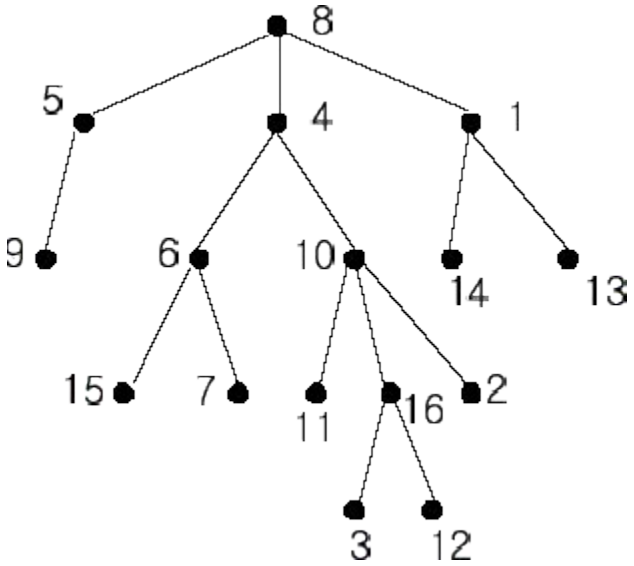
```
levelorder(root)
  q ← empty queue
  q.enqueue(root)
  while (not q.isEmpty())
    node ← q.dequeue()
    visit(node)
    if (node.left ≠ null)
      q.enqueue(node.left)
    if (node.right ≠ null)
      q.enqueue(node.right)
```



손으로 풀어보자!

손으로 순회해보자!

- 보고서에 그림, 답 포함!



재귀를 제거해보자!

재귀를 반복문으로!: Pre

- stack, queue를 이용하면 반복문으로 만들 수 있음

```
iterativePreorder(node)
  if (node = null)
    return
  s ← empty stack
  s.push(node)
  while (not s.isEmpty())
    node ← s.pop()
    visit(node)
    if (node.right ≠ null)
      s.push(node.right)
    if (node.left ≠ null)
      s.push(node.left)
```

재귀를 반복문으로!: In

- stack, queue를 이용하면 반복문으로 만들 수 있음

```
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
```

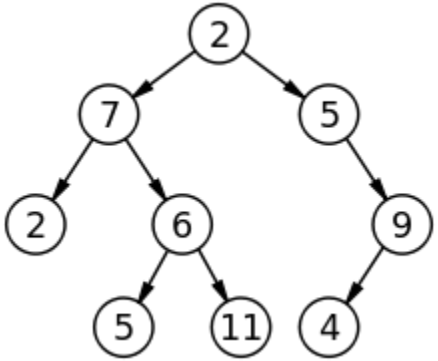
재귀를 반복문으로!: Post

- stack, queue를 이용하면 반복문으로 만들 수 있음

```
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()
```

실행 결과

- 순회 결과를 출력




```
@ Start Pre Order @
2 None
7 None
2 None
6 None
5 None
11 None
5 None
9 None
4 None
@ End Pre Order @
```

```
@ Start In Order @
2 None
7 None
5 None
6 None
11 None
2 None
5 None
4 None
9 None
@ End In Order @
```

```
@ Start Post Order @
2 None
5 None
11 None
6 None
7 None
4 None
9 None
5 None
2 None
@ End Post Order @
```

```
@ Start Level Order @
2 None
7 None
5 None
2 None
6 None
9 None
5 None
11 None
4 None
@ End Level Order @
```

테스트 코드

- ‘key’를 list로 저장하여 반환
  - 방문할 때 key를 저장하면 됨

```
75      print(node.get_key(), node.get_value())
76      result.append(node.get_key())
```

```
3   from node import Node
4   from tree_travelsal import print_pre2, print_in2, print_post2, print_level
5
6
7   class TreeTravelTests(unittest.TestCase):
8       tree = Node(key=2)
9
10      tree.set_left(Node(key=7))
11      tree.set_right(Node(key=5))
12
13      tree.get_left().set_left(Node(key=2))
14      tree.get_left().set_right(Node(key=6))
15      tree.get_right().set_right(Node(key=9))
16
17      tree.get_left().get_right().set_left(Node(key=5))
18      tree.get_left().get_right().set_right(Node(key=11))
19      tree.get_right().get_right().set_left(Node(key=4))
20
21      def test_pre_order(self):
22          result = print_pre2(self.tree)
23          self.assertEqual(result, [2, 7, 2, 6, 5, 11, 5, 9, 4])
24
25      def test_in_order(self):
26          result = print_in2(self.tree)
27          self.assertEqual(result, [2, 7, 5, 6, 11, 2, 5, 4, 9])
28
29      def test_post_order(self):
30          result = print_post2(self.tree)
31          self.assertEqual(result, [2, 5, 11, 6, 7, 4, 9, 5, 2])
32
33      def test_level_order(self):
34          result = print_level(self.tree)
35          self.assertEqual(result, [2, 7, 5, 2, 6, 9, 5, 11, 4])
36
```

기타 유용한 정보

과제) 2개!

1. 손으로 순회 그려보기(실습 트리와는 다름! 이진 트리가 아님을 염두에 둘 것)
2. 재귀가 제거된(반복문으로 된) Pre/In/Post-order travelsal 구현

실습 숙제 제출

- 숙제 제출 기한: 2018. 11. 07. 23:59:59
  - 실습 전 날
- 파일 제목: AL_학번_이름_08.zip
  - 파일 제목 다를 시 채점 안 합니다.
  - .egg 안 됨!

실습 숙제 제출할 것

- 2가지 파일을 제출
- AL_학번_이름_숙제번호.zip
  - 소스코드
    - Pycharm을 사용했을 경우 Project 디렉터리에 .idea, venv 같은 디렉터리는 제외
    - Jupyter + IPython을 사용했을 경우 ‘File - Download as’ 에서 .py 다운로드 가능
  - AL_학번_이름_숙제번호.pdf
    - 보고서는 무조건 .pdf
    - .hwp, .doc 등 채점 안 함

실습 보고서에 들어가야할 것

- 목표(할 일)
- 과제를 해결하는 방법
  - 알아야 할 것
- 과제를 해결한 방법
  - 주요 소스코드: 굳이 소스코드 전체를 붙일 필요는 없음
- 결과화면
  - 결과화면 설명(해석), 테스트코드 통과
- 보고서는 기본적으로 '내가 숙제를 했음'을 보이는 것
  - 지나치게 대충 작성하면 의심하게 됨

출석부 및 실습 점수가 궁금하다면?

- 출석부 및 실습 채점표
  - 수업 시작 후 30분까지 지각, 이후 결석
  - 실습 딜레이 1일당: -2점
    - 딜레이 2일까지: -2
    - 이후 -1씩 추가
- 튜터의 테스트 결과

질문이 생기면?

- 이름: 문현수
- 전공: 통신및보안
- 과정: 석박사통합과정 8학기
- 연구실: 데이터네트워크연구실(공5633)
- 메일: munhyunsu@cs-cnu.org
- 알고리즘은 함께 해결해가는 과목이므로 과감하게 연락
- 이메일로 처리가 안 되는 급한일: 문자/전화 등