2018년도 가을학기 SCSC 알고리즘
실습 8주차

문현수, munhyunsu@cs-cnu.org

Thursday November 01, 2018

1 개요

트리(Tree)는 컴퓨터공학에서 부모-자식 관계로 '연결되어있는' 데이터를 표현할 때 가장 널리 사용되는 자료형이다. '노드(Node)'와 '에지(Edge)'로 구성되는 트리는 그래프(Graph)의 일종이라고 볼 수 있다. 그래프(Graph)에서 상위 노드(부모)로 가는 에지나 순환이 없으면 트리로 표현할 수 있다.

그렇다면 트리는 어디에 어떻게 활용해야할까? 사용할 장소나 상황을 고려하기 전에 활용하는 방법을 먼저 학습하자. 이번 실습과 과제는 트리를 활용하기 위해 '순회(Traversal)'를 배운다.

2 트리(Tree)

트리와 그래프에서 노드는 '키(Key)'값과 '밸류(Value)'값을 가진다. 또한 에지는 시작과 끝을 가지고 있다. 이것을 구현할 때에는 노드와 에지를 각각 클래스로 정의하여 객체를 생성하는 방법과 노드 클래스에 에지 정보를 포함하는 방법이 있다. 이번 실습과 과제에서는 구현의 편의성을 위하여 노드 클래스에 에지 정보를 포함하도록 한다.

Listing 1: 에지 정보를 포함하는 Node 클래스 node.py

```
1 class Node(object):
2     def __init__(self, key=None, value=None, left=None,
          right=None):
3         self._key = key
4         self._value = value
5         self._left = left
6         self._right = right
7
```

```
8       def set_key(self, key):
9           self._key = key
10
11      def get_key(self):
12          return self._key
13
14      def set_value(self, value):
15          self._value = value
16
17      def get_value(self):
18          return self._value
19
20      def set_left(self, left):
21          self._left = left
22
23      def get_left(self):
24          return self._left
25
26      def set_right(self, right):
27          self._right = right
28
29      def get_right(self):
30          return self._right
31
32      def __str__(self):
33          return str({'key': self._key, 'value': self.
                _value,
34                      'left': self._left, 'right': self.
                           _right})
```

위의 코드는 최대 2개의 에지를 가질 수 있는 노드 클래스다. 만약 에지의 양이 정해져있지 않은 트리를 구성해야한다면 `set()`이나 `list()`를 이용하여 구성할 수 있을 것이다.

구현한 노드 클래스를 이용하여 Figure 1와 같은 트리를 구성해보도록 하자.

Listing 2: 2개의 에지를 가지는 노드 클래스를 활용한 트리

```
1   def main():
2       tree = Node(key=2)
3
4       tree.set_left(Node(key=7))
5       tree.set_right(Node(key=5))
```

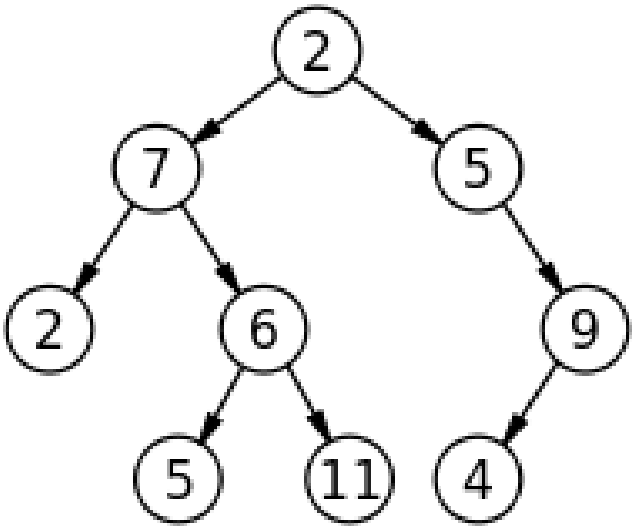

Figure 1: 이진 트리.png

```
6
7       tree.get_left().set_left(Node(key=2))
8       tree.get_left().set_right(Node(key=6))
9       tree.get_right().set_right(Node(key=9))
10
11      tree.get_left().get_right().set_left(Node(key=5))
12      tree.get_left().get_right().set_right(Node(key=11))
13      tree.get_right().get_right().set_left(Node(key=4))
14
15      print(tree)
16
17
18  if __name__ == '__main__':
19      main()
```

이진 트리(Binary Tree)는 트리 중에서도 모든 노드의 에지가 최대 2개인 트리를 말한다. 트리를 활용할 때 가장 보편적으로 사용하고 이해하기 쉬운 트리이기 때문에 실습과 과제에서는 Figure 1를 활용한다.

3 트리 순회(Tree Traversal)

트리를 활용하기 위해서는 각 노드에 접근하는 방법을 알아야한다. 트리는 하나의 루트(Root)로 시작하여 여러개의 잎(Leaf) 노드로 구성되기 때문에 일정한 규칙으로 순회하기 위한 알고리즘이 존재한다. 대표적인 트리 순회 알고리즘은 **Pre-**

order, **Inorder**, **Postorder**, 그리고 **Levelorder**가 있다. 이중에서 가장 많이 사용되는 것은 Preorder, Inorder, Postorder다.

Preorder는 트리를 순회할 때 '부모를 먼저' 처리하는 방법이다. Inorder는 '자식중 절반'을 처리한 후 부모를 처리한다. Postorder는 '자식을 모두 처리'한 후 부모를 처리한다.

트리 순회 그리고 Levelorder

Listing 3: 재귀를 활용한 Preorder

```
1  def print_pre(tree):
2      print(tree.get_key(), tree.get_value())
3      if tree.get_left() is not None:
4          print_pre(tree.get_left())
5      if tree.get_right() is not None:
6          print_pre(tree.get_right())
7
8
9  def print_in(tree):
10     if tree.get_left() is not None:
11         print_in(tree.get_left())
12     print(tree.get_key(), tree.get_value())
13     if tree.get_right() is not None:
14         print_in(tree.get_right())
15
16
17 def print_post(tree):
18     if tree.get_left() is not None:
19         print_post(tree.get_left())
20     if tree.get_right() is not None:
21         print_post(tree.get_right())
22     print(tree.get_key(), tree.get_value())
23
24
25 def print_level(tree):
26     node_queue = [tree]
27     result = list()
28
29     while len(node_queue) > 0:
30         node = node_queue.pop(0)
31         print(node.get_key(), node.get_value())
32         result.append(node.get_key())
33         if node.get_left() is not None:
```

```
34                 node_queue.append(node.get_left())
35             if node.get_right() is not None:
36                 node_queue.append(node.get_right())
37
38         return result
```

Figure **??**에 대하여 위의 순회를 수행해보자.

Listing 4: 순회 결과

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

지금까지 우리는 재귀를 활용하여 트리 순회를 살펴보았다. 이번에 해결할 과제는 재귀를 활용하지 않고 반복문을 이용하여 트리 순회를 구현하는 것이다. 그 전에 Figure 2를 손으로 순회하며 어떤 결과가 나와야하는지 살펴보자.

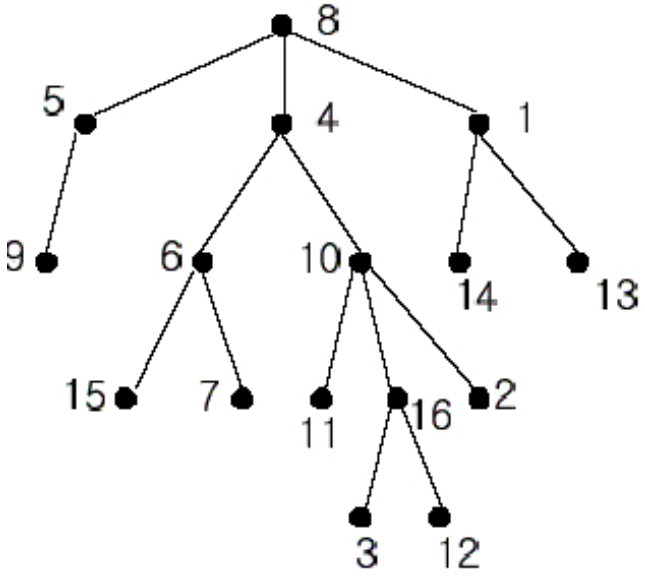

Figure 2: 트리.jpg

재귀를 이용하여 트리를 순회하면 '처라'하는 노드의 키값을 저장하기 쉽다. 노드를 처리하는 순서대로 `list()`에 입력하여 반환해주도록 하자. 이것은 수행 검증을 위한 아래 테스트 코드에서 활용할 것이다. 테스트코드를 보고 각 순회 함수의 이름을 맞추어두자(모든 순회뒤에 2를 붙인다.).

트리 순회 테스트

Listing 5: 재귀를 활용한 Preorder

```
1  import unittest
2
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
```

```
37
38 if __name__ == '__main__':
39     unittest.main(verbosity=2)
```

3.1 과제 목표

- 이진 트리가 아닌 트리 손으로 Preorder, Inorder, Postorder 순회
- 재귀를 사용하지 않고 반복문을 활용하여 Preorder, Inorder, Postorder 구현

3.2 제출 관련

- 마감 날짜: 2018. 11. 07. 23:59:59
- 딜레이: 1일당 10% 감점(처음 2일까지는 -2)
- 제출 방법: 과목 사이버캠퍼스
- 제출 형식: 과제 리포트 PDF(HWP, DOC 받지 않음!), 소스코드(구현한 .py 만 추가할 것)를 압축한 .zip 파일
- 리포트에 포함해야하는 내용: 목표, 목표를 위해 알아야하는 것, 해결 방법, 결과, (선택)느낀점 or전달할 말
- 제출 파일 제목: AL_201550320_문현수_08.zip(파일명 준수!)

3.3 조교 연락처

- 문현수
- munhyunsu@cs-cnu.org
- 공학5호관 633호 데이터네트워크연구실
- 이메일, 연구실 방문