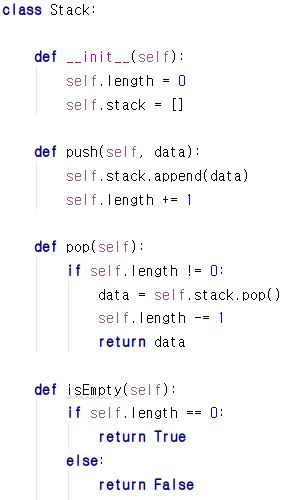
IE260 Homework #6 Report

20150359 Chiyoon Park

1. Code Description

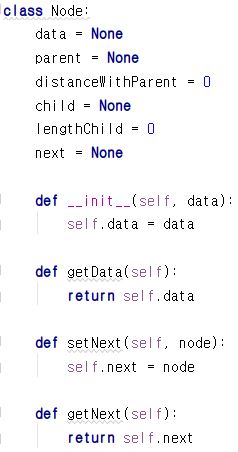
(1) Stack.py / Class Stack



Stack.py 파일에 구현한 stack 자료구조입니다. 파이썬에 내장된 List를 사용하여 구현하였습니다. 메소드는 push( ), pop( ), isEmpty( ) 세 개가 있습니다. Push는 데이터를 stack에 삽입하는 용도이며, pop은 스택의 맨 위 데이터를 제거하는 메소드입니다. isEmpty는 스택이 비었는지 아닌지를 체크합니다.

Stack 메소드는 Tree 클래스의 Find( ) 메소드에서 사용됩니다.

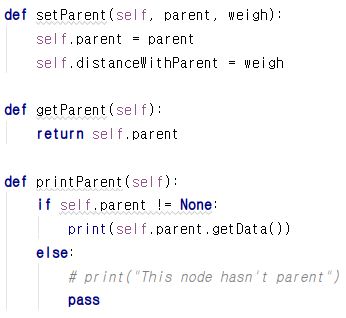
(2) Tree2.py / Class Node



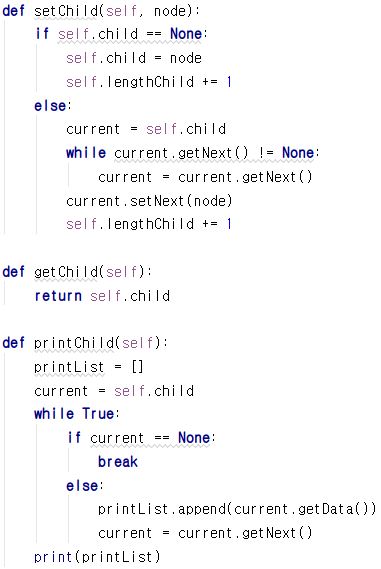
Tree2.py 파일에 구현한 Node 클래스입니다. 이 노드는 Tree의 노드의 역할을 하면서도 LinkedList의 노드로도 사용되는 클래스입니다. 안에는 data, parent, distaceWithParent, child, lengthChild, next 총 6개의 변수가 존재합니다.

먼저 \_\_init\_\_ 메소드를 통해서 parameter로 data를 입력 받습니다. 이 Homework에 대해서는 data는 노선도의 역 이름이 들어갑니다. getData( ) 메소드를 통해서 data를 받을 수 있습니다.

setNext와 getNext는 linkedlist의 노드로 사용될 때 사용하는 메소드입니다. Next로 연결된 형제 노드를 setNext와 getNext의 메소드를 통해서 다룰 수 있습니다.

****

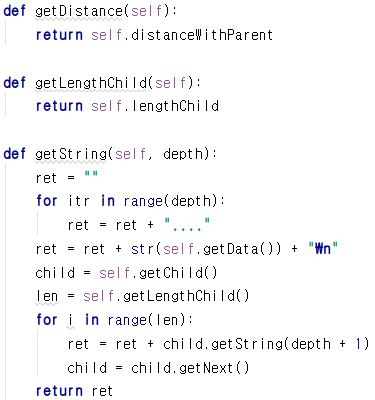
Parent는 자신의 부모 노드를 저장하는 변수입니다. setParent를 통해서 데이터를 저장하며, 이때 부모 노드와의 거리를 weigh의 parameter로 받아 distanceWithParent에 저장합니다. getParent를 통해서 parent 변수를 받을 수 있으며, printParent 메소드를 통해서 print 할 수 있습니다.



Child 는 해당 노드의 자손 노드를 담고 있는 변수입니다. 이때 child들은 linkedlist의 형태로 저장됩니다. Child에 첫 번째 자손이 저장되고, 그 노드의 형제 노드들은 next에 저장되어 linkedlist의 구조를 이루고있습니다.

setChild를 통해서 child로 저장합니다. 만약 child 변수가 비어있는 경우 child에 입력받은 node를 저장하며, 만약 이미 child가 있는 경우 linkedlist의 가장 마지막에 node를 삽입하여 저장합니다.

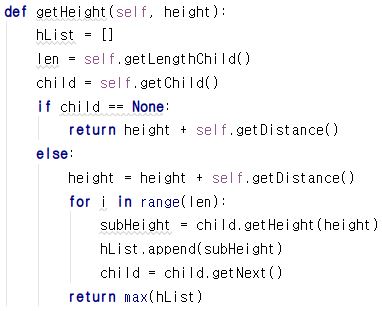
getChild를 통해서 child를 얻을 수 있습니다. printChild 메소드는 linkedlist의 형태로 이루어져 있는 child를 전부 탐색하여 List의 형태로 저장하여 반환하는 리스트입니다.



distanceWithParent 변수는 부모 노드와의 그래프상의 거리를 저장하는 변수입니다. 이를 통해서 그래프의 height를 쉽게 구할 수 있습니다.

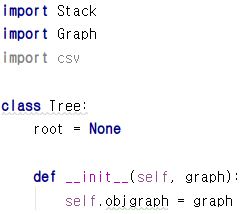
getLengthChild는 해당 노드의 자손의 개수가 몇 개 있는가 반환하는 메소드입니다.

getString 메소드는 해당 노드를 root로 하는 tree가 어떤 형태로 되어있는지 문자열로 얻어내는 메소드입니다. Recursion을 이용하여 구현하였습니다. Depth를 parameter로 받아 depth에 따라 “….”을 그려 트리를 가시적으로 드러냅니다.

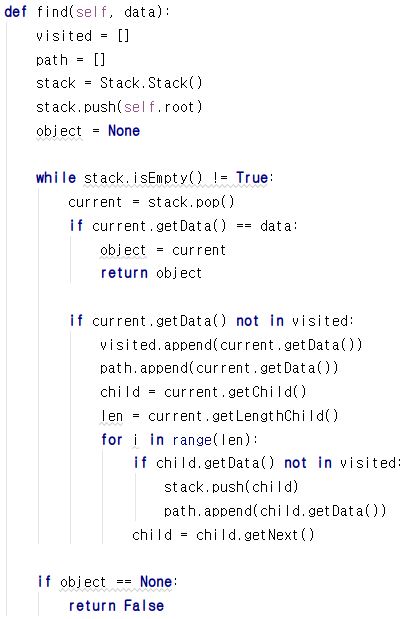


getHeight 메소드는 해당 노드 아래 height가 얼마나 있는가를 얻어내는 메소드입니다. Recursion을 통해서 노드의 child들을 탐색합니다. 이후 각 child 밑의 height가 얼마인지 확인하여 가장 높은 height를 반환합니다.

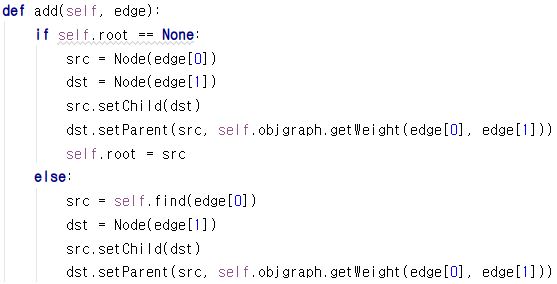
(3) Tree2.py / Class Tree



Tree 클래스는 tree 자료 구조를 만드는 클래스입니다. 기본적으로 root 변수가 존재하고, graph를 기반으로 tree를 만들기에 해당 graph를 parameter로 입력받아 objgraph 변수에 저장합니다.

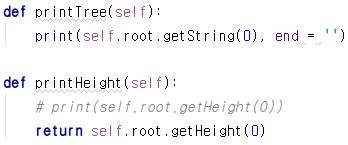


Find 함수는 data를 입력하면 해당 data를 가지고 있는 노드를 찾아 반환하는 메소드입니다. 해당 메소드는 DFS 알고리즘을 통해서 tree를 탐색하고, 전부 탐색하여 만약 data를 가진 노드를 찾는다면 이를 return하며, 만약 존재하지 않을 경우 False를 반환합니다.



Add 함수는 Edge를 확인하여 해당 edge의 destination을 source에 자손으로 설정하는 메소드입니다. Parameter를 통해서 Edge를 입력받습니다. 만약 root가 없는, 즉 트리 안에 노드가 하나도 없는 경우 처음 입력 받은 Edge의 source를 tree의 root로 설정합니다. 그리고 dst를 src의 자손 노드로 설정합니다. 이후 dst 노드의 부모 노드와의 거리를 objGraph에서 얻어 입력합니다.

만약 root가 이미 있는 경우 해당 edge를 읽어 src와 dst를 얻어낸 후 src를 find를 통해서 찾아내어 해당 노드의child로 dst를 삽입합니다.



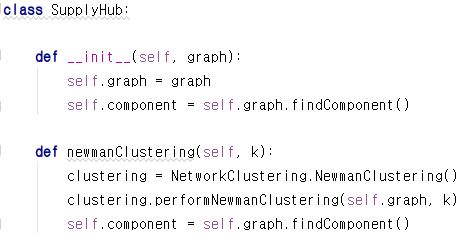
printTree는 root 노드에 getString 메소드를 실행하여 그래프를 그려내는 메소드입니다.

printHeight의 경우 root 노드로부터 모든 자손 노드를 확인하면서 height를 계산하는 메소드입니다.

Child

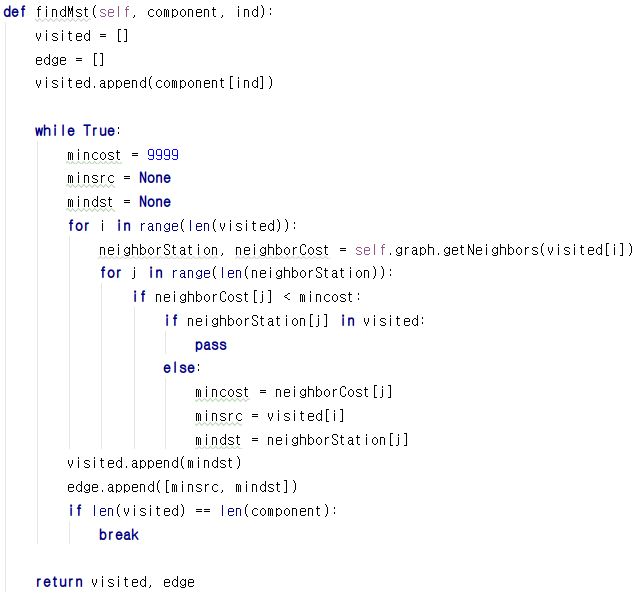
위의 Tree 클래스를 그림으로 표현하면 다음과 같이 나타낼 수 있습니다. 노드와 노드는 부모와 자식 관계로 이어지며, 자손들끼리는 linkedlist의 자료 구조로 되어있습니다.

(4) hw6.py / Class SupplyHub



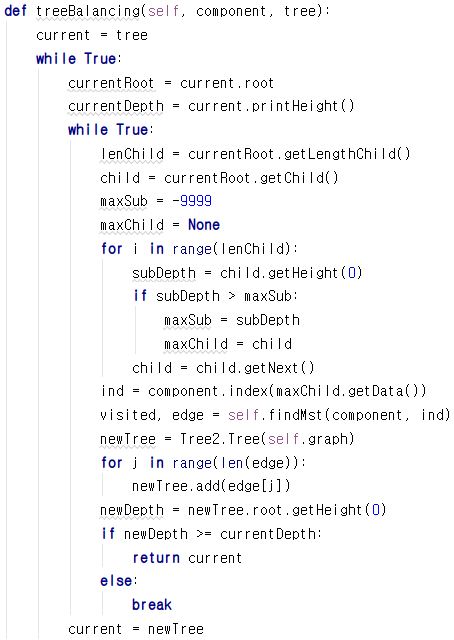
SupplyHub는 지하철 노선도를 분석하여 효과적인 supply hub의 위치를 찾는 클래스입니다. 먼저 \_\_init\_\_ 메소드를 통해서 parameter로 그래프를 입력 받습니다. 이후 해당 그래프의 component를 Graph 클래스의 findComponen( )를 통해서 분석하여 self.component에 저장합니다.

newmanClustering 메소드는 parameter로 입력받은 k개 만큼의 클러스터를 생성하는 메소드입니다. NetworkClustering.py 파일에 있는 NewmanClustring( ) 메소드를 통해서 k개의 클러스터를 형성한 후, self.component에 얻어낸 component를 저장합니다.

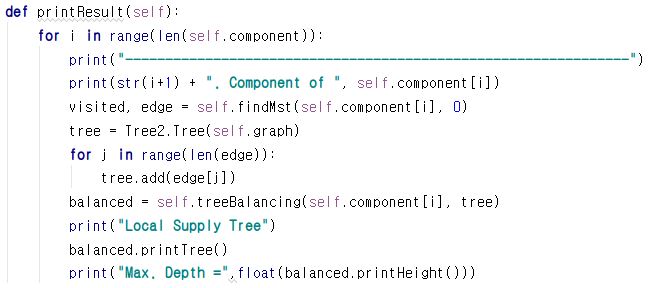


findMst 메소드는 parameter로 component 중 하나를 리스트의 형식으로, component의 어떤 위치에서 MST를 형성할지를 ind로 받아 Minimum spanning tree를 형성하는 메소드입니다. Prim’s algorithm을 사용하여 ind로 입력 받은 정수가 가르키는 component의 원소로부터 그래프를 탐색합니다. Visited에 방문한 vertex를 기록하고, edge에 방문한 edge를 순차적으로 기록합니다. 이후 그래프를 탐색하면서 아직 방문하지 않은 edge를 가장 weigh가 적은 순서대로 방문합니다.

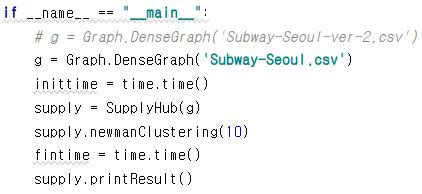
이후 구해진 visited와 edge를 반환합니다.



Tree balancing 메소드는 parameter로 입력받은 component로 형성한 tree를 자식들의 height가 최대한 비슷하게 balancing하는 메소드입니다. 두 개의 while문을 통해서 동작하는데, 처음 입력받은 tree의 모든 자손들을 탐색하여 가장 height가 큰 자손을 찾습니다. 이후 그 child를 기준점으로 component에 대해서 MST를 진행하여 새로운 Tree를 만듭니다. 이후 그 tree의 height를 비교하여 계속 tree의 최대 깊이를 작게 만들어갑니다. 그리고 생성한 트리를 반환합니다.



printResult 메소드는 위의 메소드들을 사용하여 최종적으로 supply hub를 설치할 역을 찾고, 각 component의 트리 구조와, 최다 깊이를 프린트하는 메소드입니다. for문을 통해서 모든 component를 탐색하면서 findMst( ) 를 진행하고 tree를 balancing합니다. 최종적으로 형성된 tree를 출력하고, 해당 tree의 깊이를 구합니다.



위에 구현한 메소드들을 실행하는 부분입니다.

2. Result

Subway-Seoul-ver-2.csv 파일로 분석한 결과입니다. 원본 파일의 결과는 txt로 첨부하였습니다.

---------------------------------------------------------------

1. Component of ['소요산', '동두천', '보산', '동두천중앙', '지행', '덕정', '덕계', '양주', '녹양', '가능', '의정부', '회룡', '망월사', '도봉산', '도봉', '방학']

Local Supply Tree

덕계

....양주

........녹양

............가능

................의정부

....................회룡

........................망월사

............................도봉산

................................도봉

....................................방학

....덕정

........지행

............동두천중앙

................보산

....................동두천

........................소요산

Max. Depth = 24.0

---------------------------------------------------------------

2. Component of ['창동', '녹천', '월계', '광운대', '석계', '신이문', '외대앞', '회기', '청량리', '제기동', '신설동', '동묘앞', '동대문', '종로5가', '종로3가', '종각']

Local Supply Tree

회기

....외대앞

........신이문

............석계

................광운대

....................월계

........................녹천

............................창동

....청량리

........제기동

............신설동

................동묘앞

....................동대문

........................종로5가

............................종로3가

................................종각

Max. Depth = 16.0

---------------------------------------------------------------

3. Component of ['시청', '서울역', '남영', '용산', '노량진', '대방', '신길', '영등포', '신도림', '문래', '영등포구청', '당산', '합정', '홍대입구', '신촌', '이대', '아현', '충정로']

Local Supply Tree

남영

....용산

........노량진

............대방

................신길

....................영등포

........................신도림

............................문래

................................영등포구청

....................................당산

....서울역

........시청

............충정로

................아현

....................이대

........................신촌

............................홍대입구

................................합정

Max. Depth = 20.0

---------------------------------------------------------------

4. Component of ['구로', '구일', '개봉', '오류동', '온수', '역곡', '소사', '가산디지털단지', '독산', '금천구청', '광명', '석수', '관악', '안양']

Local Supply Tree

구로

....구일

........개봉

............오류동

................온수

....................역곡

........................소사

....가산디지털단지

........독산

............금천구청

................석수

....................관악

........................안양

................광명

Max. Depth = 20.0

---------------------------------------------------------------

5. Component of ['부천', '중동', '송내', '부개', '부평', '백운', '동암', '간석', '주안', '도화', '제물포', '도원', '동인천', '인천']

Local Supply Tree

동암

....백운

........부평

............부개

................송내

....................중동

........................부천

....간석

........주안

............도화

................제물포

....................도원

........................동인천

............................인천

Max. Depth = 17.0

---------------------------------------------------------------

6. Component of ['명학', '금정', '군포', '당정', '의왕', '성균관대', '화서', '수원', '세류', '병점', '서동탄', '세마', '오산대', '오산']

Local Supply Tree

수원

....화서

........성균관대

............의왕

................당정

....................군포

........................금정

............................명학

....세류

........병점

............서동탄

............세마

................오산대

....................오산

Max. Depth = 20.0

---------------------------------------------------------------

7. Component of ['진위', '송탄', '서정리', '지제', '평택', '성환', '직산', '두정', '천안', '봉명', '쌍용', '아산', '배방', '온양온천', '신창']

Local Supply Tree

직산

....두정

........천안

............봉명

................쌍용

....................아산

........................배방

............................온양온천

................................신창

....성환

........평택

............지제

................서정리

....................송탄

........................진위

Max. Depth = 26.0

---------------------------------------------------------------

8. Component of ['을지로입구', '을지로3가', '을지로4가', '동대문역사문화공원', '신당', '상왕십리', '왕십리', '한양대', '뚝섬', '성수', '건대입구', '구의']

Local Supply Tree

왕십리

....상왕십리

........신당

............동대문역사문화공원

................을지로4가

....................을지로3가

........................을지로입구

....한양대

........뚝섬

............성수

................건대입구

....................구의

Max. Depth = 12.0

---------------------------------------------------------------

9. Component of ['강변', '잠실나루', '잠실', '신천', '종합운동장', '삼성', '선릉', '역삼', '강남', '교대', '서초', '방배', '사당']

Local Supply Tree

선릉

....역삼

........강남

............교대

................서초

....................방배

........................사당

....삼성

........종합운동장

............신천

................잠실

....................잠실나루

........................강변

Max. Depth = 13.0

---------------------------------------------------------------

10. Component of ['낙성대', '서울대입구', '봉천', '신림', '신대방', '구로디지털단지', '대림']

Local Supply Tree

신대방

....신림

........봉천

............서울대입구

................낙성대

....구로디지털단지

........대림

Max. Depth = 9.0