2019 – 1 자료구조 과제3

과 목 명 : 자료구조
과 제 명 : 지하철 길 찾기 외 1 개
사용언어 : 파이썬(jupyter notebook)
학 과 : 심리학과
학 번 : 20141963
이 름 : 박승완
제 작 일 : 20190607
담당교수 : 김승태교수님

지하철 빠른 길 찾기

- **기능 :** 서울/수도권 지하철 노선도를 가져온다.
  - 양이 너무 많으니 개인적으로 정리해서 일부만 가져온다.
  - 지하철 역은 30개가 넘지 않도록 한다. 주로 환승역을 이용한다.
  - 예) 2호선과 그 내부의 연결된 다른 호선 및 환승역 30개를 이용한다.
  - 역 사이의 거리를 넣는다. 현실에 기반하지 않아도 된다.
  - 출발점과 도착점을 입력하면 경로와 최단거리를 표시한다.
  - 강의용 슬라이드의 코드는 최단거리만 표시하며, 경로는 응용하는 방법을 찾아야 합니다.

    실행 예)

    모든 역을 표시한다. (BFS, DFS를 이용. 배열에서 바로 출력하지 말것) : 역 이름이 표시되어야 출발역, 도착역 이름을 입력할 수 있음

    출발역 : *사당*

    도착역 : *선정릉*

    사당 – 교대 – 선릉 – 선정릉 (5.5km)
- **기능 :**

출발역에서 도착역까지의 소요시간과 경로를 나타낸다.

필요한 것 :

메서드 : bfs(처음에 이용 가능한 역을 보여주기 위해 역들을 Bfs로 탐색), visitStation(S집합에 넣기)

지하철들과 다음 역까지의 소요시간이 표현되어 있는 딕셔너리(stations)

기능구현의 핵심 :

메인함수에서

1 소요시간이 가장 짧은 역부터 긴 역 순으로 방문하고 방문한 역은 visited을 1로 초기화하여 표시한다.

2 새로운 역을 방문하면 그 역과 이어진 역들까지의 소요시간을 갱신한다. 단, 이전에 이미 최소 소요시간이 구해졌다면 소요시간을 서로 비교하여 작은 것으로 바꾸거나 유지한다.

visitStation메서드는 현재 역과 연결된 역들과의 시간을 계산하고 시간이 더 짧은 것으로 업데이트를 한다.

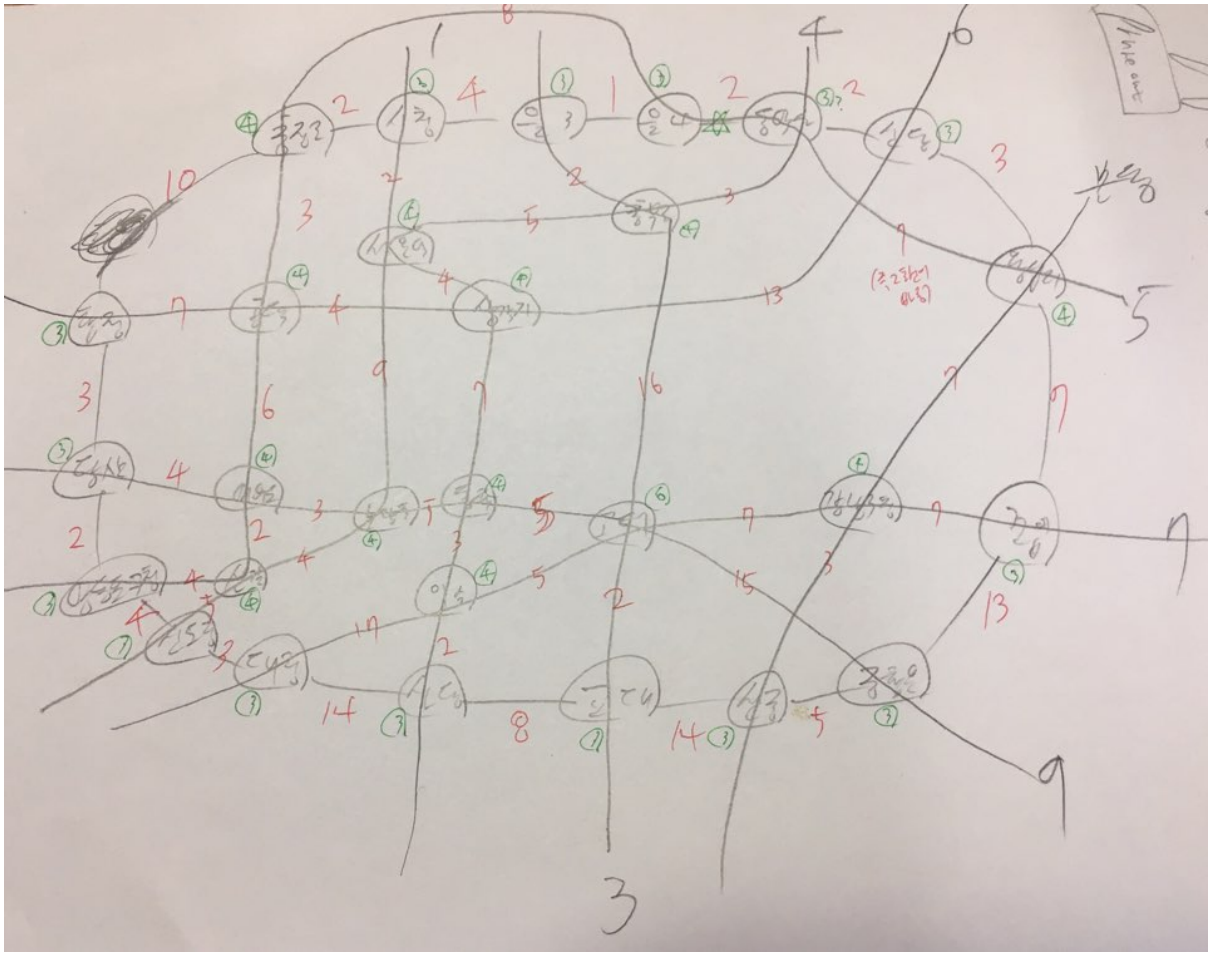
3 그래프의 모든 역들을 방문할 때 까지 1, 2과정을 반복한다.

추가정보 : jupyter notebook에서 코딩하였습니다. 확장자도 .ipynb입니다.

.py로도 저장할 수 있는 기능을 알아내어 .py도 첨부하겠습니다!

교수님께서는 거리를 예시로 드셨지만 네이버 검색결과에 따른 **소요시간**으로 만들어서 실제 **활용할 수 있는 프로그램**입니다.

지하철 노선도 :



- **결과 표시 :**

```
==========[지하철 노선도]==========
이용가능한 역:  ['충정로', '합정', '시청', '공덕', '을지로4가', '당산', '을지로3가', '서울역', '삼각지', '여의
도', '동대문역사문화공원', '영등포구청', '충무로', '노량진', '신당', '동작', '신길', '신도림', '고속터미널역', '왕
십리', '이수', '대림', '강남구청', '종합운동장', '교대', '건대입구', '사당', '선릉']
출발지를 입력하세요:
사당
도착지를 입력하세요:
선릉
========

 [ 사당 -> 선릉 ]
경로 :  ['사당', '이수', '고속터미널역', '강남구청', '선릉']
소요시간 :  17 분이 소요됩니다.
```

사전 해싱

- **문제 내용** : **51840 개의 단어로 이루어진 단어 파일이 제공된다.**
  - 단어 파일은 단어와 뜻으로 이루어져 있다.
    - 예) apple : n. 사과
  - 단어 파일은 정렬되어 있지 않다. 단어 파일을 가져와 메모리에 담아둔다. 파일 입출력 방법을 미리 알아야 한다.
  - 단어를 입력하면 뜻을 표시하는 프로그램을 작성한다.
  - 단어 사전은 이진 탐색 트리와 해싱의 2가지 기법을 이용한다. 프로그램 안에는 2 개의 방식이 모두 있어야 한다.(선택이 아님)
    - 2가지 기법(탐색트리, 해싱)의 코드가 소스에 포함되어 있어야 한다.
- **실행 예** :

  단어 : *apple*

  n. 사과 (13) <- 이진 탐색 트리를 통해 찾은 단어

  n. 사과 (6) <- 해싱 테이블을 통해 찾은 단어

  뒤의 숫자는 이 단어를 찾기 위해 이진 탐색 트리에서 비교를 몇 번 했는지, 또는 해싱 테이블 또는 단어사전에 몇 번 접근했는지를 표시한다.

- **해결 방안** :

52000, 70000, 80000개의 해시테이블의 크기를 이용하여 해시 탐색을 할 수 있는 프로그램을 만들었다. (크기는 수동으로 입력해야한다.)
단어와 의미를 따로 리스트로 저장하여 이용한다.
찾는 단어가 없으면 이를 표시하기도 한다. (오류 처리)
파일에서 문자열들을 가져오는 함수, 해싱을 위해 단어들을 세팅하는 함수, 해시키를 구하는 함수, 해시키를 이용하여 해시 테이블에 넣는 함수, 해시키를 이용하여 뜻을 찾는 함수, 이분 탐색을 하는 함수로 구성된다.

분석 :
해시테이블의 크기가 작으면 단어 리스트 후반부로 갈 수록 충돌이 많이 나는 경향을 보였다. 그래서 이분탐색보다 결과가 좋지 않은 경우도 많았다. 이분 탐색은 평균 10~15회의 검색 횟수를 보였지만 해시 탐색은 0에서 100이 넘어가기도 했다.

해시 테이블의 크기가 충분히 클 경우에는 이분탐색보다 월등히 적은 탐색 횟수를 보인다. 대부분이 10회 이하, 특히 0~4 사이에서 나타난다.
그렇지만 상대적으로 시간이 더 걸린다.

추가정보 : jupyter notebook에서 코딩하였습니다. 확장자도 .ipynb입니다.
.py로도 저장할 수 있는 기능을 알아내어 .py도 첨부하겠습니다!

- **결과 표시** :

```
단어 : apple

binary search
n.사과 ( 14 )

hash search
n.사과  ( 5 )
```

<실제 출력 화면>

```
52000
apple 14, 5
roi 15, 0
longline 15, 0
ran 8 ,1
toothpaste  14, 17
didymous 15, 26
rhombus 14, 102


70000
apple 14, 0
roi 15, 0
longline 15, 0
ran 8 ,1
toothpaste 14, 2
didymous 15, 7
rhombus 14, 12


80000
~
이미 충분히 작기에 생략
~
toothpaste 14,0
didymous 15, 6
rhombus 14, 1
```

<각각의 해시 테이블 크기에 따른 검색 결과>

(단어, 이분 탐색 시도 횟수, 해싱 충돌 횟수)

감사합니다.