과제 #4

M3239.002300_003 데이터 사이언스 특강

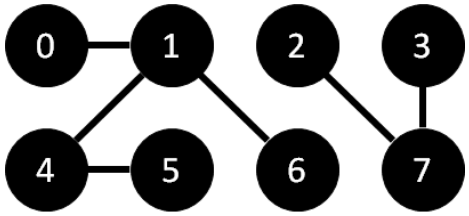
Due: 2021년 10월 27일 23시 59분

1 문제 1: Node간 거리 확인

강의 시간에 배운 breadth-first search 알고리즘을 활용하여 graph 사이의 두 node 간의 거리를 구하는 알고리즘을 구현하고자 한다. 이를 위해 class Graph와 method check_distance를 구현하라. class 생성자 및 method 정의는 아래와 같다.

- **class Graph(filename)**

  filename 파일에 적힌 edge 정보를 읽어와 undirected graph를 구성한다. Graph의 각 node를 표현하기 위해 0 이상의 정수가 한 개 씩 ID로 배정되며, filename 파일에는 graph의 모든 edge 정보가 적혀 있다. 예를 들어 아래 그림과 같은 그래프의 경우,



  아래 예시의 small.txt 파일과 같이 표현된다.

```
spdsta@login:~/HW4$ cat small.txt
0,1
1,4
4,5
1,6
2,7
3,7
```

입력 파일에는 한 줄에 edge 한 개씩 x,y 형태로 기입되어 있다. 파일 안에는 edge가 중복 없이 기입되어 있다고 가정한다. 예를 들어 0,1이 파일 안에 두 번 나타나거나, 0,1과 1,0 둘 다 파일에 기입되어 있는 경우는 없다고 가정해도 된다.

- **check_distance(x, y, max_distance)**

  x를 ID로 가지는 node와 y를 ID로 가지는 node 사이의 거리가 max_distance 보다 작거나 같은지의 여부를 확인하여 반환한다. x에서 y로 가는 path가 없으면 max_distance값과 상관 없이 False를 반환한다. 아래는 실행 예시이다.

```
spdsta@login:~/HW4$ python3 -i prob1.py
>>> graph = Graph('small.txt')
>>> print(graph.check_distance(0, 1, 1))
True
>>> print(graph.check_distance(0, 1, 2))
True
>>> print(graph.check_distance(0, 4, 1))
False
>>> print(graph.check_distance(0, 4, 2))
True
>>> print(graph.check_distance(0, 2, 1))
False
>>> print(graph.check_distance(0, 2, 1000000))
False
```

  x, y, max_distance 모두 int 타입이며, x, y로는 항상 유효한 ID (입력 그래프 파일 안에 한 번 이상 등장한 ID)가 들어온다고 가정한다. 또한 x와 y는 항상 다르다고 가정한다. max_distance로는 1 이상의 정수가 들어온다고 가정한다.

이번 과제는 알고리즘의 실행 시간 또한 채점 기준에 포함된다. 지나치게 비효율적으로 알고리즘을 구현하여 실행 시간이 정해진 기준보다 오래 걸릴 경우 점수가 감점되거나 0점 처리될 수 있다. 구체적인 내용은 아래와 같다.

- 성능 측정은 실습 서버를 기준으로 한다.
- 성능 확인시에는 Stanford Large Network Dataset Collection에서 제공하는 Facebook Large Page-Page Network 데이터셋을 사용한다[1]. 해당 데이터셋은 22,470개의 node, 171,002개의 edge로 구성되어 있다. 실습 계정 홈 디렉토리의 HW4 디렉토리에 large.txt 파일로 해당 그래프가 저장되어 있다.
- HW4 디렉토리에 성능 테스트를 위한 python script (test.py)가 저장되어 있다. 제출 전 테스트 스크립트를 사용하여 성능에 따른 감점 여부를 확인할 수 있다.

[1] https://snap.stanford.edu/data/facebook-large-page-page-network.html

아래는 성능 테스트를 통과한 경우의 출력 예시이다.

```
spdsta@login:~/HW4$ python3 test.py
Pass (3.308 sec)
```

아래는 성능 테스트를 통과하지 못한 경우의 출력 예시이다. 이 경우 성능에 따라 감점이 있을 수 있다.

```
spdsta@login:~/HW4$ python3 test.py
Fail (33.266 sec)
```

2 구현 및 제출

실습 서버의 실습 계정 홈 디렉토리의 HW4 디렉토리에 prob1.py를 작성하여 저장한다.

파일	설명
prob1.py	Graph class 구현

제출 시 아래와 같은 사항을 유의하라.

- 제출 파일에는 **반드시 class를 정의하는 코드만 포함되어야 하며, 테스트 코드 등이 포함되어서는 안 된다**. 과제 채점 시 실행 예시 스크린샷에서 보인 바와 유사한 방법으로 기계적인 채점을 수행할 예정이다. 이 때 제출한 코드에 불필요한 내용이 포함되어 채점이 방해될 경우 해당 과제가 0점 처리될 수 있다.
- 이번 과제의 경우 **강의 예제에서 다룬 모듈 외에 다른 모듈을 import하여 사용하는 것은 허용하지 않는다.** 예를 들어 collection module은 사용 가능하나, threading module은 사용할 수 없다.
- 채점 시 성능 테스트 스크립트와 유사한 테스트셋을 사용하여 성능을 확인한다. 성능 테스트 스크립트를 통과하는 구현의 경우 성능 감점이 없도록 보수적으로 테스트셋 및 실행 시간 기준을 설정할 예정이다.
- 성능 테스트 결과가 수강생 중 가장 빠른 경우 추가 점수가 있을 수 있다.
- 실습 서버의 HW4 디렉토리에 파일을 저장하면, 과제 제출 시간에 자동으로 해당 파일을 복사해 간다. 파일명을 다른 이름으로 수정할 경우 (예: p1.py) 과제가 제출되지 않을 수 있으니 유의하여야 한다.
- 그레이스 데이를 사용할 경우 spdsta@aces.snu.ac.kr로 이를 알려야 하며, 메일 제목은 [SPDS] HW4-Graceday 학번 이름의 포맷으로 작성하라 (예: [SPDS] HW4-Graceday 2020-20000 안규수).