Homework 5

SNU L0444.200, 2017 Fall

Due: 12/17 (Sun), 23:59

TA: Jinyung Kim, Prof: Kwangkeun Yi

이번 마지막 숙제에서는 스마트폰 시대에 쉽게 접할 수 있는 개념인 위치기반 데이터를 바탕으로, 새롭게 방문해 볼 만한 곳을 추천하고, 새로운 길을 제안해 주는 프로그램을 만들어본다.[1]

그간 강의와 실습과 숙제를 통해 배운 내용을 동원해 본다.

- (지금까지 과제로 만들던 것보다는) 큰 프로그램 작성. 필요하다면 부품들을 만들고 조립식이나 계층쌓기로 생각하여 문제를 해결해 본다.
- 소프트웨어적 문제해결에 동원되는 개념과 실습한 내용 적용. 필요한 부분은 어디이며 필요한 개념은 어떤 것인지 생각해 보고, 적재적소에 활용해 문제를 해결해 본다.
- 그간의 숙제 재사용. 가능하다면 지난 숙제들에서 만들어 본 함수들을 부품으로 활용해 문제를 해결해 본다.

네 개의 함수를 작성해 보게 된다.

- Exercise 1, 2, 3: 만들 프로그램의 핵심 기능을 담당할 함수들을 작성해 본다.
- Exercise 4: 기능은 동일하지만, 조금 더 일반적인 경우에도 올바르고 빠르게 동작하도록 확장해 보기 위한 하나의 방법을 고민해 본다.

[1] 2017년 봄 컴퓨터과학이 여는 세계 (SNU 046.016) 수업에서 박미현, 이도경, 김연두 학생이 제출한 '누구나 모험가가 될 수 있는 모험 소프트웨어 제안'을 참고하여 각색함

아래와 같은 자료형들을 사용한다.

'위도'와 '경도' 위도는 -90 초과 90 미만, 경도는 -180 초과 180 미만의 값이다. 이번 문제에서는, 실수값인 위도와 경도를 다루기 쉽도록 1,000,000을 곱한 후에 소수점 아래를 버린 정수값으로 표현한다. 예를 들어 서울대입구역의 위치는 대략적으로 위도 37.481236, 경도 126.952733이므로, 이번 문제에서 사용하는 자료형으로는 위도는 37481236, 경도는 126952733으로 표현한다.

'시각' 양의 정수값으로 시각을 표현한다. (Unix time: 1970년 1월 1일 00:00:00 UTC로부터 몇 초가 지났는지) 예를 들어, 한국시각으로 2017년 1월 1일 자정은 1483196400으로 표현한다.

'위치정보' 표현 이름과 키워드들의 리스트가 기록된 위치정보를, 다음과 같은 길이 4인 튜플을 이용해 표현한다. 모든 장소는 이름이 서로 다르며, 빈 문자열은 아니라고 가정한다. (`Tuple[int, int, string, list[string]]`)

- 0번째 값: '위도'를 나타내는 정수값 (`int`)
- 1번째 값: '경도'를 나타내는 정수값 (`int`)
- 2번째 값: 이 장소의 이름을 나타내는 고유한 문자열 (`string`)
- 3번째 값: 이 장소를 표현하는 키워드들을 나타내는 문자열들의 리스트 (`list[string]`)

사용자의 '특정 시간동안의 위치' 표현 사용자가 특정 시간 동안 있었던 위치를 표현하기 위해, 다음과 같은 길이가 3인 튜플을 이용한다. (`Tuple[int, int, string]`)

- 0번째 값: 이 장소에 있었던 시작 '시각' (`int`)
- 1번째 값: 이 장소에 있었던 마지막 '시각' (`int`)
- 2번째 값: 이 장소의 이름 (`string`)

사용자의 '시간별 위치' 표현 '특정 시간동안의 위치' 들의 리스트로 사용자의 시간별 위치와 이동 정보를 표현한다. 리스트 내에서는 이동한 순으로 기록한다. 다음 위치의 시작 시각은 이전 위치의 마지막 시각 이후이다. (`List[Tuple[int, int, string]]`)

Exercise 1 “일상적인 곳, 특별한 곳”

새로운 곳을 추천해 주기 위해, 사용자가 방문한 곳을 잘 알아보자.

사용자의 ‘시간별 위치’가 주어진다. 이를 바탕으로 이 사용자가 방문한 장소들 중 일상적인 곳과 특별한 곳의 목록을 구하는 함수 `common_uncommon(lst, n)`를 작성해 본다. `lst`는 ‘시간별 위치’로, 길이는 500 이하이다. `n`은 정수이다. `n`회 이상 방문한 곳은 일상적인 곳으로, 1회만 방문한 곳은 특별한 곳으로 정의한다. 함수가 리턴하는 것은 길이 2인 `string` 리스트들의 리스트이다. 0번째 값은 일상적인 곳의 이름들의 리스트이다. 1번째 값은 특별한 곳의 이름들의 리스트이다.

각 리스트는 다음과 같이 정렬한다. 일상적인 곳은: 방문 횟수가 많은 곳부터. 방문 횟수가 같다면, 해당 위치에 있었던 시간의 총 합이 큰 것부터. 그것도 같다면, 가장 먼저 방문한 곳부터. 특별한 곳은: 가장 먼저 방문한 곳부터.

Exercise 2 “비슷한 새로운 곳 제안”

사용자가 방문한 곳과 비슷하면서도 근처에 있는 곳도 방문하도록 추천해 보자.

사용자의 ‘시간별 위치’가 주어진다. 길이는 500 이하이다. 주변의 ‘위치정보’들의 리스트도 함께 주어진다. 길이는 5000 이하이다. ‘시간별 위치’에 포함된 모든 위치들은 ‘위치정보’들의 리스트에도 들어있다고 가정해도 된다. 즉, ‘시간별 위치’에서 사용된 모든 장소의 이름을 가진 위치가 ‘위치정보’들의 리스트에 들어있다는 것이다.

사용자가 방문한 곳들 중, ‘특별한 곳’들, 즉 단 한 번씩만 방문한 곳들에 대해, 근처에 있으면서도 이와 비슷한 곳들을 구하는 함수 `suggest_similar(lst, loc_lst, d, k)`를 작성해 본다. `lst`는 ’시간별 위치’이다. `loc_lst`는 ‘위치정보’들의 리스트이다. `d`는 기준이 될 거리를 나타내는 실수이다. `k`는 기준이 될 유사도를 나타내는 0 이상 1 이하의 실수이다. 함수가 리턴하는 것은 `string`들의 리스트이다. 제안해 줄 곳들의 이름들의 리스트이다. 순서는 임의의 순서여도 되며, 아래의 조건을 만족하는 곳들의 이름만을 모두 포함하고 있으면 된다.

- 주변의 ‘위치정보’들의 리스트에 있으면서,
- 사용자가 한 번도 방문하지 않은 곳이면서,
- 사용자가 방문한 ‘특별한 곳’ 중 적어도 하나와는 거리가 `d` 미만이면서,
- 사용자가 방문한 ‘특별한 곳’ 중 적어도 하나와는 키워드를 기반으로 한 유사도가 `k` 이상인 곳.

거리는 지금까지와 마찬가지로 `ctp17hw1` 라이브러리를 이용한다. 이번 숙제에서 사용된 위도와 경도 자료형을 원래의 실수값으로 되돌려 호출해야 함에 유의한다. 키워드를 기반으로 한 유사도는 숙제 4의 문제 4의 정의와 같다.

Exercise 3 "전혀 다른 길 제안"

사용자가 방문한 곳과는 전혀 다른 새로운 곳을 지날 수 있는 길을 제안해 보자.

사용자의 '시간별 위치'가 주어진다. 길이는 500 이하이다. 주변의 '위치정보'들의 리스트도 함께 주어진다. 길이는 5000 이하이다. '시간별 위치'에 포함된 모든 위치들은 '위치정보'들의 리스트에도 들어있다고 가정해도 된다. 즉, '시간별 위치'에서 사용된 모든 장소의 이름을 가진 위치가 '위치정보'들의 리스트에 들어있다는 것이다.

사용자가 방문한 곳들 중, '일상적인 곳'과 '일상적인 곳' 사이에 이동하면서 들릴 수 있을 만하면서도, 지금까지 방문한 곳들과는 전혀 다른 곳들을 구하는 함수 `suggest_unusual(lst, loc_lst, n, t, r, k)`를 작성해 본다. `lst`는 '시간별 위치'이다. `loc_lst`는 '위치정보'들의 리스트이다. `n`은 '일상적인 곳'을 판단할 때 기준이 되는 정수이다. `t`는 기준이 될 시각을 나타낼 정수이다. `r`은 기준이 될 거리의 비율을 나타내는 1보다 큰 실수이다. `k`는 기준이 될 유사도를 나타내는 0 이상 1 이하의 실수이다. 함수가 리턴하는 것은 `string`들의 리스트이다. 제안해 줄 곳들의 이름들의 리스트이다. 순서는 임의의 순서여도 되며, 아래의 조건을 만족하는 곳들의 이름만을 모두 포함하고 있으면 된다.

- 주변의 '위치정보'들의 리스트에 있으면서,
- 사용자의 '시간별 위치'에서, 연속된 두 '일상적인 곳' (사용자가 `n`회 이상 방문한 곳) 사이에 들를 만한 곳. 단, 두 연속된 '일상적인 곳' 사이의 시각의 차이가 `t` 이상일 때만 고려한다. 즉, 이전에 방문한 곳의 마지막 시각과, 다음에 방문한 곳의 시작 시각의 차가 `t`이상일 때만.
  - 두 곳 사이에 새로운 곳까지 들렀을 때의 총 거리가, 두 곳 사이의 (아무 곳도 들르지 않았을 때의) 거리의 `r`배 이하인 곳.
  - 사용자가 방문한 그 어떤 곳과도 유사도가 `k` 이하인 곳.

거리는 지금까지와 마찬가지로 `ctp17hw1` 라이브러리를 이용한다. 이번 숙제에서 사용된 위도와 경도 자료형을 원래의 실수값으로 되돌려 호출해야 함에 유의한다. 키워드를 기반으로 한 유사도는 숙제 4의 문제 4의 정의와 같다.

Exercise 4 "위치정보도, 처리할 경우도 많다면"

앞의 세 문제를 통해 누구나 모험가가 될 수 있도록 새로운 곳과 길을 제안해 주는 프로그램들을 작성해 보았다. 이를 일반적인 경우로 확장할 수 있으려면, 입력으로 받는 데이터의 크기가 매우 클 때에도, 또한 처리해야 할 경우가 많을 때에도 올바르고 빠르게 동작할 수 있게끔 해야 할 것이다. 이를 위해 다음과 같은 하나의 경우에 한정해서, 문제를 확장해 보도록 하자.

주변의 '위치정보'들의 리스트가 주어지면, 특정한 위치로부터 거리가 `d` 미만이면서, 가장 가까운 다른 위치의 이름을 구하는 프로그램을 작성해 본다. '위치정보'들의 리스트는 하나만 주어지지만, 가장 가까운 위치를 알고 싶은 곳들은 여러 곳이 주어진다. 각각에 대해서 조건에 맞는 위치를 구하면 된다.

함수 `nearest(lst, name_lst, d)`를 작성해 본다. `lst`는 주변의 '위치정보'들의 리스트로, 길이는 500,000 이하이다. `name_lst`는 가장 가까운 다른 위치를 알고 싶은 위치의 이름들의 리스트로, 길이는 10,000 이하이다. 모든 이름은 '위치정보'들의 리스트 안에 들어있다고 가정해도 된다. 함수가 리턴하는 것은 위치의 이름들의 리스트이다. `name_lst`에 주어진 순서대로, 각 위치와 거리가 `d` 미만이면서, 가장 가까운 다른 위치의 이름들의 리스트를 만들어 돌려주면 된다. 거리가 `d` 미만인 위치가 없으면 빈 문자열로 한다.

거리는 지금까지와 마찬가지로 `ctp17hw1` 라이브러리를 이용한다. 이번 숙제에서 사용된 위도와 경도 자료형을 원래의 실수값으로 되돌려 호출해야 함에 유의한다.

모든 경우를 일일히 고려하는 것보다는 실행비용의 측면에서 효율적인 프로그램을 작성해야, 큰 입력에 대해서도 빠른 시간 내에 맞는 답을 낼 수 있을 것이다.