Project #2: Reachable Cities

Due: January 22nd, 2023 at 23:59

Note on Collaboration: You may work with other students. However, each student should write up and hand in his or her project separately. Be sure to indicate with whom you have worked in the comments of your submission.

Problem statement

The objective of this project is to write a Python program that uses a database to find all the cities reachable from a given city in a given number of steps.

A city c' is reachable from a city c in k steps if there is a sequence of cities $(c_0, c_1, ..., c_k)$ that satisfies the following conditions:

1. $c_0 = c$ and $c_k = c'$
2. all the c_i's are different
3. c_{i+1} is in the neighborhood $N(c_i)$ of c_i $(i = 0, ..., k-1)$. $N(c_i)$ is defined as the set of cities that are
   - in the same province as c_i or
   - on the same river or lake as c_i or
   - on the same sea but at a distance less than s or
   - anywhere at a distance less than d

   where s and d are given parameters and the distance between c_i and c_j is defined as

   $$|c_i.latitude - c_j.latitude| + |c_i.longitude - c_j.longitude|$$

4. no shorter sequence satisfies conditions 1, 2, and 3.

Input: a city name n and a country code c (this is necessary because several cities may have the same name), an integer k (the maximum number of steps), two real numbers d and s (the maximum distance and the maximum distance on seas), and the *mondial* database that contains information about cities, countries, lakes, and rivers.

Output:

a list of all cities reachable from n in 1, 2, ..., k steps.

Example: For the input $n =$ "`Geneva`" , $c =$ "`CH`", $k = 5$, $s = 4$, $d = 2$ the output should look like

```
1 Step
Bern (CH), Villeurbanne (F), Besancon (F), Turin (I),
...
2 Steps
```

```
Monaco (MC), Nancy (F), Nice (F), Milan (I), Metz (F),
...
3 Steps
Vaduz (FL), Ajaccio (F), Genua (I), Marseille (F), Strasbourg (F),
...
4 Steps
Wiesbaden (D), Schaffhausen (CH), Leverkusen (D), Clermont Ferrand (F),
...
5 Steps
Frankfurt am Main (D), Tunis (TN), Lille (F), Heidelberg (D),
...
```

Indications

1. The following tables of the Mondial database are relevant for this application: City, Country, located
2. You can start by writing a Python function

   `neighbors(city: str, country: str, d: float, s: float) -> set[(str,str)]` that queries the mondial database to find the neighbors of a city located in a country. The function should return a set of pairs (city, country).
3. Use a breadth-first search to find the cities at distance 1, then 2, then 3, etc.
4. To query the Mondial database you can use the following code template:

```
from urllib.request import urlopen
from urllib.parse import quote, urlencode
...
...
q = << construction of a string with your sql query >>
eq = quote(q)
url = "http://kr.unige.ch/phpmyadmin/query.php?db=mondial&sql="+eq
query_results = urlopen(url)
# iterate over the result rows
for line  in query_results :
    string_line = line.decode('utf-8').rstrip()
    # if the query had a syntax error or if the result is empty
    # there is nevertheless one empty line, ignore it
    if len(string_line) > 0:
        # put the column values in columns
        columns = string_line.split("\t")
        << do something with the values >>
query_results.close()
```