HOMEWORK ASSIGNMENT #2 – RANKING

MANAGEMENT OF BIG WEB DATA (FALL 2022-23)

GENERAL INSTRUCTIONS

- The work can be done in pairs or in singles.
- For the purpose of submission, your **username** will be <first_name>.<last_name>, or <first_name_of_member1>.<last_name_of _member1>.<first_name_of_member2>.<last_name_of _member2> for a pair. E.g., alice.smith or alice.smith.bob.walker
- Your solution should be submitted in a single zip file named **<username>-hw1.zip**, through Moodle. Only one member of the pair should submit.
- The zip will include an answers document in pdf format, named **<username>-hw1.pdf** make sure to write your name(s) on the top. The answers can be written in English or Hebrew.
- For every question that requires writing code, code should be written in python version 3.XX and submitted in a **.py** file. Keep the code readable, as it will be graded.
- You are expected to submit an original work. Copying from references (e.g. existing crawlers) or looking at others' solutions is not acceptable. Do not publish your solution or save it in any public place (specifically, public GitHub projects) – in cases of cheating, all involved students will be equally held liable.

TASKS

In this exercise, all the code will be submitted in a single file, **ex2.py.**

You can use libraries for randomness and for operations on data structures, but do not use existing implementations of indexing, tf-idf, PageRank, MCMC or top-k.

1. Write a function `def myData()` which returns a "mini Web" of 8-10 websites and the links between them (of your choice – you can invent any websites and connections between them, or you can use real websites). The output will be a list of dictionaries, such that each dictionary represents a website and has
    - The key "URL" whose value is the website URL (a string). It does not have to be a real URL
    - The key "tokens" whose value is the website tokens, a list of strings in English without spaces or special characters, representing an already processed website. Tokens may be repeated more than once and may not be ordered.
    - The key "linksTo" whose value is a list of strings – URLs. The links should all point to other websites in your mini Web. Each Website can have 0 up to 7-9 links, avoid self-links.

For example:
{'URL': 'example.com', 'tokens': ['jaguar', 'mammal', 'felidae', 'family'], 'linksTo': ['example.org']}
{'URL': 'example.org', 'tokens': ['jaguar', 'cute', 'big', 'cat', 'jaguar', 'my', 'favorite'], 'linksTo': []}
{'URL': 'example.il, 'tokens': ['my', 'cat', 'big'], 'linksTo': ['example.com', 'example.org']}
Draw a diagram of the mini web you chose in the answers file.

2. Write a function `def mySearchString()` which returns a list of keywords (strings) as a search string over your "mini Web" from question 1, e.g., : ['jaguar', 'family']

3. Write a function `def invertedIndex(data, searchString)` which gets as input data in the format of question 1 and 2 and returns a dictionary. The keys of the dictionary will be all the tokens in `searchString`, and the value for each token will be a list of lists. Each inner list will include two values, a URL and its tf-idf score with respect to this token and document represented by the URL. The lists should be ordered by descending tf-idf score.

   Write the result of `invertedIndex(myData(),mySearchString())` in the answers file (add line breaks for readability).

   For example:
   {'jaguar': [['example.com', 0.05], ['example.com',0.04]], 'family': [['example.com', 0.07]]}

4. Write a function `def pageRankSimulation(data, numIter, beta)` which gets as input data in the format of question 1 and two parameters and computes the PageRank of each URL in your "mini Web" using a random surfer simulation, executed for numIter steps using the input beta parameter as the damping factor. The output is a list of lists. Each inner list will include a URL and its computed rank.

   Write the result of `pageRankSimulation(myData(),100000,0.8)` in the answers file (add line breaks for readability). Does the result match your intuition on what is the "most important" website? Explain. Note that the result may differ between executions do to the randomness of the simulation process.

   For example:
   [['example.com', 0.28], ['example.org',0.52], ['example.edu',0.2]]

5. Write a function `def score(tfIdf, pageRankValue)` which gets as input two numbers representing the tf-idf score and PageRank of some website, and produces an overall score. The function must be **monotone and reasonable in the context of ranking websites for search – but do not use simple sum or average**. Explain the function you chose in the answers file.

6. Write a function `def top1(invertedIndex, pageRank)` which gets as input data in the format of the output of questions 3-4 and computes the top-1 website using the function `score()` from question 5, and either TA or FA. The function should print each access it makes (e.g., “random access to example.com at the PageRank index”) and return the URL of the top-1 website and its score. Your implementation does not have to support efficient random access, nor to be generalizable to any number of columns and any k.

   Write the result of `top1(invertedIndex(myData(),mySearchString()),pageRankSimulation(myData(),100000,0.8))` in the answers file. Was the Website your algorithm selected also the most relevant/important? Did the algorithm you chose to implement save accesses compared with naively going over the relevant indices? Explain briefly