Homework 4 : MapReduce

Course: Distributed Systems Monsoon 2021, IIIT-H

Due Date: 1st November, 2021, 10 PM

Introduction:

We will be using the Map-Reduce framework using Hadoop streaming. You are expected to implement Mapper and Runner components in Python language, with the runner script that calls these in a language you would prefer (Python, C++, Java, or a bash script).

You will be working in teams, the same ones that will be for your project.

The job scheduling instructions for a streaming job will be provided in your mail with the credentials for server access.

Questions:

Q1:

Given a large file of words, with one word on each line, rearrange the words so as to find anagrams. (An anagram of a word, is another word with the same letters (even number of each letter)).

The output should be a set of lines where each line has the anagram (with the letters sorted in alphabetical order) first, followed by a colon ":", and all the words that are the anagrams in the input file. Please refer to the example for more understanding.

(Optional: You can arrange the keys into a data structure so that one can search on the keywords. *Not Bonus*)

Example:

Input:

ant
tan
eat
tea
name
mane

Output:

Ant: ant, tan

Aemn: mane, name

Aet: eat, tea

Q2:

Given an URL and an integer K as an input, recursively crawl through the URLs and retrieve the URLs referred in each webpage, which further are scraped. Using this information you will create a directed graph, where the nodes are the unique URLs visited in your scraping, and an edge exists from node U to node V, if on scraping U, you find a link to V. (i.e., U refers V).

Please note you don't have to exhaustively find all the links (ie., some links are embedded etc., even a one pass "href" collector is okay to use.) But you will have to get a considerable amount of links from the page. Because it will be important for Q3 too. (You won't be graded for getting ALL the links, but have to get a fair amount of them)

The depth K will be ~25 (not strict, if it is too time consuming, we will consider K=15 too. The code working for a large K isn't a requirement for grading, depending on whether it works appropriately grades will be given) and the input URL will be to a relatively simple website. (Not like Pizza Hut or Amazon with a lot of links)

Example:

(not real, just for understanding)

Input:

https://courses.iiit.ac.in/my/ 2

Expected Run:

On scraping https://courses.iiit.ac.in/my/ (lets call it O) (iteration 1), among other urls, you will find a reference to a course C. You will also find a reference to an assignment A of the course C. On recursively scraping for another iteration (iteration 2),

- When you scrape A, you will find a reference to C (as it is the course of that assignment),
- And on scraping C, you will find a reference to A (as it is an assignment in the course).

Therefore a part of the graph here will be (listing out the edges from Node X to Node Y):

O C

O A

A C

C A

There will be more edges other than this, but this is for understanding of what output is expected.

The **output** will be a text file with the first line being two integers, Number of Nodes (URLS) and Number of Edges (unique references, a webpage may be referred more than once, but it should only be counted once), with a whitespace between them:

10 34

O C

O A

A C

C A

…. 30 more lines denoting references

Q3:

Using the output graph from Q2 as an input to this question, find the connected components in the graph. You will be required to give the number of connected components, and then proceed to give the Nodes/URLs in each component in each new line. (Component of Size 1 is possible)

Any algorithm is fine, either you can read and use from existing sources, or develop some algorithm on your own.

The graph input to this question will always be:

2 <= Number of Nodes <= 100

1 <= Number of Edges <= 1000,

Irrespective of what output Q2 can give in some cases. Your code will only be tested on the above constraints.

Input:

3 3

1 2

2 3

3 1

Output:

1

1 2 3

Submission Details:

The submission should be done in the form of a zip file (Naming convention: <Team id>.tar.gz) .

The directory structure should be such:

• Directories (Naming convention: <Question id>/) for each question, with the respective source codes (Naming convention: <Question id>.<file_extension>) in those .

• A README.md to briefly explain your code and algorithm.

Please strictly adhere to the expected format and handle appropriate edge cases, or you will be penalized.

Plagiarism is strictly prohibited (between students, or from the net), and when found strict actions will be taken.