HW2

Deadline: Mar. 14th 2:59 P.M. (before class)

In previous homework, you have practiced basic Spark RDD operations (transformations and actions) with word count example. In this homework, you will need to implement more complicated operations with a real-world input file. The input to the program are pages from the English-language edition of Wikipedia. You will generate an adjacency list for each page.

(Definition of adjacency list: https://en.wikipedia.org/wiki/Adjacency_list)

Data Description

The wiki-micro.txt on Blackboard are a sample version of pre-processed Wikipedia corpus in which the pages are stored in an XML format, with many thousands of pages per file. This has been further preprocessed such that all the data for a single page is on the same line.

Each page of Wikipedia is represented in XML as follows:

```
<title>Page_Name</title>
(other fields that we don't care about)
<revision optionalAttr="val">
   <text optionalAttr="val">
      (page body)
   <\text>
<\revision>
```

As mentioned before, the pages have been "flattened" to be represented on a single line. So this will be laid out on a single line like:

```
<title>Page_Name</title> (other fields that we don't care about) <revision optionalAttr="val"> <text optionalAttr="val"> (page body) <\text> <\revision>
```

The body text of the page also has all newlines converted to spaces to ensure it stays on one line in this representation.

TODO

Retrieve all the Wikipedia internal links for each page and generate an adjacency list.

In the body text, the internal links are enclosed by double square brackets. In an example of "Granderson played college baseball at the [[University of Illinois at Chicago]]", [[University of Illinois at Chicago]] is a link that can direct you to the UIC page on Wikipedia.

In addition, your program must meet the following requirements:

1. all the page names should be in lowercase
2. if a page contains multiple identical links, only one should be recorded (i.e. only take unique values)
3. the final output should be sorted by the number of links in a descending order. For example, if page_a contains 5 unique links and page_b contains 8 unique links, page_b should be outputted first.

The output should be in the following format if you use `.saveAsTextFile()` from the example.

('page_a', a list of unique links from 'page_a')
('page_b', a list of unique links from 'page_b')
('page_c', a list of unique links from 'page_c')
…

Submission

Python code and output file (both saved in plain txt file.)