WikiCrawler

Our algorithm related to crawler consists of several parts:
getWebPageByUrl function:

This function return page by URL, but before doing that check politeness policy, i.e for each 100 request to the server wait 5 seconds. It also show error in the case that we face with network error. Finally, it return the page as a string.
The pseudo code for This code is:

- Check Number of request
    - For every 100 request wait for 5 sec.
- Create input stream and output stream.
- Get URL by OpenStream.
- If an error occur in Getting URL, show the error in console.

Crawl function

This is a simple BFS function with queue. We add a list to each page in order to don't report the duplicate link in a page.
Pseudo Code for Crawl:

- Create a new file/ use previous file.
- Write the Max number of requested node in the File
- While (Queue is not empty)
    - SeedURL= Queue. Dequeue
    - list.add(SeedURL)
    - Call mycrawl2
    - list.clear()
- Close the file

MyCrawl2 function:

This function get seedURL which set by BFS and crawl that page. Then, it check politeness policy, i.e. don't send request to disallowed website from robot.txt. Finally, retrieve all of Link in the page and then call insideCrawler to check the raw file for each of them.

Pseudo Code for MyCrawl2:

Use Variable PVisited for Count number of node
Use Variable Visited for check the max number of running of program in worst case

- If This URL is rejected by robot.txt
    - Return
- If Number of Visited is equal to max
    - Return
- Else If Visited contain URL
    - Duplicate case, so Return.
- Else
    - Visited.add URL
- If PVisited.size<max
    - PVisited.add URL
- Else If !PVisited.contain(URL)
    - Return(the case that we should finish)
- HTMLPage = getWebPageByUrl(BASE_URL + seedUrl);
- findHref = "<a href=";
- findP = "<P>";
- Find firstp with findP variable and remove the first part from HTMLPage
- Find the FirstHref
- While(We have Href)
    - Check that Href is not contain any :or # and it start with /wiki/
    - If PVisited.size<max OR (PVisited.size>max and PVisited.contain(Href))
        - if(!Visited.Contain(HREF))
            - If(!list.contain(HREF))
                - If(InsideCrawl(HREF))
                    - PVisited.add URL
                    - List.add URL
                    - Write URL in file
                    - Queue.enqueue
        - if(Visited.Contain(HREF))
            - If(!list.contain(HREF))
                - PVisited.add URL
                - List.add URL
                - Write URL in file

InsideCrawler function:

This function give the seedURL and then get Raw file , before that check politeness policy, i.e. don't send request to disallowed website from robot.txt. Then return True (the selected keywords are in the raw file) or false (at least one of the selected keywords are not in the raw file)

Pseudo Code for InsideCrawler:

- Remove "/wiki/" from input URL
- Create Raw URL from input URL
- Get Pageby URL(Raw URL)
- If It is not satisfy the Check polite function
    - Return
- For all of Keyword
    - If One of keyword not exist in the page
        - Return False
- Return True

Politeness function

This function set the PolitePage variable by getting the robot page and set this variable

Pseudo Code for Politeness:

- String s=GetPageByURL("https://en.wikipedia.org/robots.txt")
- SetPolitePage(s)

CheckPolite function :

This function get one URL and check that if this URL not exist in disallowed website return true, so we could get page. Otherwise, return false. Because in the robot.txt for wikipedia we have just three Allowed URL. we need to check that our target URL is not in Allowed part and is in the page to return false.

- Result=True
- If URL exit in PolitPage(robot file)
    - Result=False
    - If URL is in one of 3 Allowed URL
    - Result=True
- Else
- Result=True
- Return Result

2. For each epsilon (0.01, 0.005): List webpages with top 15 page rank. How does the list change as epsilon changes?

3. For each epsilon (0.01, 0.005): Number of steps that your page rank algorithm took to converge (within epsilon)
4. For wikiTennis: Pages with top 15 page rank, top 15 indegree, and top 15 outdegree and Jaccard Similarities among these sets. Do this for epsilon 0.005
5. For your favourite topic: Web pages with top 15 page rank, top 15 indegree, ad top 15 outdegree. ReportJaccard similarities among these sets. Do this for epsilon = 0.005

Then the graph created (and stored in WikiTennisGraph.txt) should have exactly 100 vertices.

PageRank

For this section we have created 4 classes as following.

Class DGraph

This class has defined in order to read the Graph text file and ignores the first line which is the number of nodes and then create a directed graph
These are the methods of the class:

- AddNode
    - Input node as string
    - if there is no node with this name Add to the graph
- AddEdge
    - Input start and dest as 2 nodes
    - If both nodes are exist then add edge p-->q
    - mGraph.get(start).add(dest)
- edgeNumber
    - Returns the number of edges of the Graph
- graphSize:
    - Returns the size of the Graph
- nodesNameList:
    - Returns the number of nodes in the graph as an String Array
- edgesFrom
    - Returns the set as an out degrees of the specific node
- edgesTo
    - Returns the set as an out degrees of the specific node
- nodeIndex

    - Input Node Name
    - Return index of the node in the Graph

PageRank Class

This is the main class of this section. It provides all the required methods to calculate page rank and top K in-degree and out-degree. Methods of this class are as following:

- PageRank
    - Call textToGraph() method
    - Call pageRank method
    - Set P_n =<1/N,.....1/N>
    - Call NextStep function to calculate P_{n+1}
    - If Norm(P_{n+1}, P_n) <= epsilon then return P_{n+1} as a pageRank
- NextStep
    - Input P_n
    - Calculate P_{n+1} based on the algorithm in the noteBook
    - Return P_{n+1}
- inDegreeOf
    - Input node of the Graph
    - Return in-degree of this node
- outDegreeOf
    - Input node of the Graph
    - Return out-degree of this node
- pageRankOf
    - Input node of the graph
    - Returns PageRank if this node
        - From PageRank Array
- topKInDegree
    - Input k
    - Calculate pageRank
    - Create ArrayList for pageRank
    - Call outDegreeOf for each node
    - Sort ArrayList DESC (from highest to the lowest)
    - Return index from i=0 to k
- topKOutDegree
    - Input k
    - Calculate pageRank
    - Create ArrayList for pageRank
    - Call calculate outdegree for each node
    - Sort ArrayList DESC (from highest to the lowest)
    - Return index from i=0 to k
- topKPageRank

    - Input k
    - Calculate pageRank
    - Create ArrayList for pageRank
    - Sort ArrayList DESC (from highest to the lowest)
    - Return index from i=0 to k
- textToGraph
    - Read the text file from the current directory
    - Read line by line and ignore the first line
    - Split the line by space
    - Call addNode()
    - Call addEdge
- Norm
    - Input Pn and Pn+1
    - Create an array for norm
    - Return sqrt this differences

MyWikiRanker

This class creates a wiki graph based on previous section. We calculate Jacc Similarity for top 100 page ranks, top 100 in-degree and top 100 out-degree in 2 case with $\varepsilon = 0.01$ and $\varepsilon = 0.005$ and the result are as following:

Number of Steps to Converge= 3
Epsilon= 0.01
Jacc Sim(topRank,topInDegree)= 0.53
Jacc Sim(topRank,topOutDegree)= 0.24
Jacc Sim(topInDegree,topOutDegree)= 0.41

Number of Steps to Converge= 3
Epsilon= 0.005
Jacc Sim(topRank,topInDegree)= 0.53
Jacc Sim(topRank,topOutDegree)= 0.24
Jacc Sim(topInDegree,topOutDegree)= 0.41

As we can see the results are the same, so they are not dependent on epsilon.

The following results are for Top 15 pages:
Number of Steps to Converge= 3
Epsilon= 0.01
Jacc Sim(topRank,topInDegree)= 0.30
Jacc Sim(topRank,topOutDegree)= 0.11
Jacc Sim(topInDegree,topOutDegree)= 0.15

Number of Steps to Converge= 3
Epsilon= 0.005
Jacc Sim(topRank,topInDegree)= 0.30
Jacc Sim(topRank,topOutDegree)= 0.11
Jacc Sim(topInDegree,topOutDegree)= 0.15

Here also we see no difference with epsilon.

MyTennisRanker

This class creates a wiki tennis graph based on previous section. We calculate Jacc Similarity for top 100 page ranks, top 100 in-degree and top 100 out-degree in 2 case with $\varepsilon = 0.01$ and $\varepsilon = 0.005$ and the result are as following:
Number of Steps to Converge= 2
Epsilon= 0.01
Jacc Sim(topRank,topInDegree)= 0.72
Jacc Sim(topRank,topOutDegree)= 0.29
Jacc Sim(topInDegree,topOutDegree)= 0.36

Number of Steps to Converge= 3
Epsilon= 0.005
Jacc Sim(topRank,topInDegree)= 0.71
Jacc Sim(topRank,topOutDegree)= 0.29
Jacc Sim(topInDegree,topOutDegree)= 0.36

As we can see the results are very similar.

The following results are for 15 pages Top 15 pages and top 15 in-degree and out-degree:

Number of Steps to Converge= 2
Epsilon= 0.01
Jacc Sim(topRank,topInDegree)= 1.0
Jacc Sim(topRank,topOutDegree)= 0.15
Jacc Sim(topInDegree,topOutDegree)= 0.15
Number of Steps to Converge= 3
Epsilon= 0.005
Jacc Sim(topRank,topInDegree)= 0.875
Jacc Sim(topRank,topOutDegree)= 0.15
Jacc Sim(topInDegree,topOutDegree)= 0.15