1 Introduction

In this homework you will create a Java or Spark program to calculate ranks in a small model of the world wide web, using graphs. A graph is defined as $G = \{V, E\}$ where $V$ is a set of vertices or nodes and $E$ is a set of edges or links between the nodes. You need to preprocess pages in MapReduce or Spark. To solve the analytical component (discovering interesting features of $G$) you can process $G$ in Spark, MapReduce or SQL. If you decide to use SQL we will provide a Java program that computes recursive queries [1, 2], but you are responsible customizing it to solve this homework.

In essence a web site (or web page) is a vertex in a huge graph. All hyperlinks in the page are the edges in the graph, connecting it to other pages (at same site or at other vertices). Evidently these edges have a direction, since “clicking” on a link takes you away from the page you were and unless there is a link back (assuming there is no “back” button in your browser) you couldn’t return to where you were originally. This is called a “directed graph”. Notice there can be duplicate edges that must be eliminated.

The degree of a vertex is the number of edges that are incident on the vertex. In a directed graph we can divide this number into two parts: the indegree is the number of edges that are incident on the vertex and the outdegree is the number of edges that is ”pointing outward” from the vertex.

In this homework you will calculate the degree of each page. The input files will be HTML files, with embedded text and hyperlinks. The links are clearly marked as “anchors” in HTML language:

\[\text{<a href = "//test.html"> link </a>}\]

We will strictly adhere to this convention. Furthermore since the files will have the “.html” extension you will be able to visualize them in your browsers. The output will consist of several interesting features of the graph.

2 Program and output specification

The main program should be called ”AnalyzeGraph”. If you use Java this is a sample call. If you use Scala in Spark you can change to parentheses notation.

Syntax:

\[\text{AnalyzeGraph edge=<site|page>;k=<integer>}\]

where edge indicates if you are going to build a ”coarse” graph with edges between sites, or a ”fine” graph with edges between pages and $k$ is the maximum path length for exploration. Given an adequate definition you can obtain the coarse graph from the fine graph.
3 Requirements

- $G$ will be represented by a binary $E$ matrix, where $G$ does not have loops (self-cycles), but $G$ can have cycles of length 2 or more. That is you need to make sure $E$ has zeroes on the diagonal. You will store the input edge matrix $E$ (adjacency matrix) in sparse form; a dense matrix storage is unacceptable (too easy and very inefficient).

- You will build an edge $(i, j)$ with two alternatives as specified in the input, eliminating duplicate edges, adding a frequency:
  (1) between a web site and another web site (coarse)
  (2) between two web pages (fine)

- The input will be a set of HTML files in the parallel cluster stored on HDFS. This set of pages corresponds to a web crawl.

- Programming: you must use Java/MapReduce or Scala/Spark to preprocess pages to build $G$. To analyze $G$ you can use MapReduce, Spark or SQL. Given the size of $G$ and its sparsity you cannot use plain C++, Java or R.

- You need to program or call a sparse matrix multiplication algorithm. You will multiply $E \cdot k$ times to get $E^2 = E \cdot E$ and $E^3 = E^2 \cdot E$. Notice that after each multiplication you need to set the diagonal to zeroes to eliminate cycles. That is, $k$ is an input parameter. You will compute $G^+ = E + E^2 + \cdots + E^k$.

- You can assume the maximum path length $k \leq 10$.

- Based on $E$ and $G^+$ you will identify:
  (a) the longest path length
  (b) min/max in-degree and out-degree in $E$
  (c) histograms for degree distribution
  (d) the source vertex with highest degree (most "outgoing" vertex).
  (e) the destination vertex with highest degree (most popular).

- The program should not halt when encountering errors. It should just send a message to the log file and continue with the next line. The only error that is unrecoverable is a missing input file or a missing output file.

- Deliverables: source code and short document describing interesting findings on $G$ and $G^+$. You are expected to show and explain your program to the TA.

References
