Joshua Steward

Lowell Batacan

Page Rank Algorithm

In our implementation of Google's page rank algorithm, we take a set of pages represented by a directed adjacency matrix, and calculate the ranks based on pages linked to pages and the number of outgoing links in pages. It is a stage built iterative approach, and importantly, the rankings depreciate over time. Before we talk about the rankings' depreciation, we will first detail how our implementation of page rank works.

We tried several approaches to how we would iterate through stages for each page, calculating a more depreciated rank for each page in every stage. We had a hard-coded set number of runs for one of our implementations, and two different approaches for several other implementations. Inside these tests for termination, we calculated each stage of page ranking. To do this, we first iterated through each page. For each page, had an array detailing the connections going into the current page, the number of connections out of the page, and then algorithms to compute these values. In the actual ranking algorithm, we used these values to add up all the pages linking in divided by their respective link-out values, and then multiplied that by the alpha depreciation values. This simulated a primitive version of the page rank algorithm, and allowed us to test ranks against previous rankings for convergence tests. We also used fractions to represent the ranking values, as fractions display a greater level of detail, an important factor in dealing with algorithm analysis.

In terms of convergence factors, the damping agent used as the variable d should create a convergence effect on this type of stage based algorithm. If for each stage the difference in the score is taken between the current node and the last node, and the result is less than some pre-defined convergence threshold, then it shows the ranks have converged into more usable values. This is ideal in the page-rank algorithm, especially with massive amounts of data, such as the trillions of pages Google must sort through. And essentially, since each rank diverges from it starting rank, it must eventually converge if some non-random type algorithm is used for searching. So, given some indefinite amount of runs, it must eventually converge, and will generally be terminated just short of converging.