Caim LAB
Session 5: PageRank

Miquel Gómez i Esteve

Presentation

In this practice we had to implement the PageRank computation. With the pseudocode given to us it was not very hard to implement the basic algorithm.

Changes and decisions

I added some modifications while I was implementing the rest of the program.
The most noticeable change is the one explained below:

- Instead of using the edgeList and edgeHash structures, I used the routes and routesHash structures only, which I think allowed me to compute the PageRank more efficiently. That's because every airport has in its routes attribute a list of all the routes with itself as a destination, as well as the weight and an index pointing to the origin airport position in the Airport list.

By using this data structures and after many tests I can assure that the computation will be fast enough.

Issues found

The first issue I encountered when working on this practice was to read both files and store all the info the way I wanted. After many tries I was able to do it correctly.

I found the second and main issue after implementing the algorithm given to us in the practice statement. The problem was that after an iteration of the algorithm, even the first. The result stored in the variable P was not correct, because, the sum of all its values must be equal to 1. The issue was that some of the airports had outweight equal to 0, and we were losing its pagerank values.
The solution I found after thinking about it for some time was to distribute the values of these airports amongst all the values of all the airports in every single iteration.

Observations

I knew that the Page Rank algorithm converges very fast, and after some experiments I was able to see it. That means that the converge value is not that important, nevertheless a good one value I found would be 10^-10, because the result would be very similar even if it was smaller, and the time spent computing is reasonable.
After trying different values for the damping value, I observed that the lower it is the faster the algorithm converges. However, the optimal value to chose would be a very high one, because the smaller the value is, the less importance we give to the edges of the graph and the more similar the final values will be between themselves. As said in the practice statement the best values would be between 0.8 and 0.9, since the higher ones would mean a much larger execution time. The concrete value you chose would depend on if you give more importance to the velocity or the precision.
Apart from the converge stopping criteria I also added a maximum number of iterations.