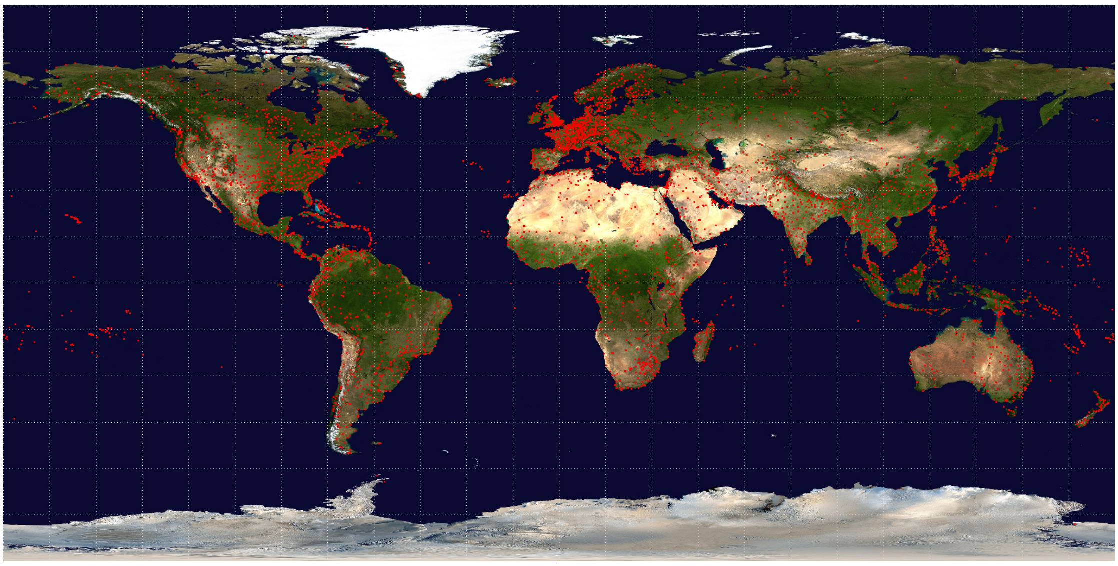
Information Retrieval
Lab 03 - Page-Rank

Asaf Badouh
Laura Cebollero Ruiz

November 2018



1 Introduction

In this lab we want to compute the page rank of airports using the network defined by routes from and to airports. For that, we used the supplied data downloaded from *Open flights*[1]:

- `airport.txt` - contains a list of airports (nodes) from the world.
- `routes.txt` - contains a list of routes (edges) from the world.

As explained in the lab, we can represent this information as a graph, where airports are nodes and routes are edges. The weights of the edges will be the number of flights between two airports. Airports with high PageRank value are *important* airports. This is crucial information when designing and reforming airport as it means that many flights (with passengers and goods) pass it. Optimizing the operation of such airport, will lead to improvement of many supply chains and other visitors.

2 Implementation Scheme

2.1 Data Structures

While analyzing `airport.txt` we found $7,663$ airport records. $1,923$ of them don't have IATA code, 2 of them appear twice which leave us with $5,738$ "valid" airports. Representing the problem in two-dimensional array (matrix) will consume $2^{25} \times sizeof(datatype)$ bytes. Since the max weight is 534 (ORD - Chicago Ohare Intl, United States) our datatype must be at least 2-bytes, i.e. total of 64 Mega-Bytes. Since the matrix is very sparse, in order to save data space[2], we decides to save the information in two dictionaries, one for each, airport and routes (see figure.1)

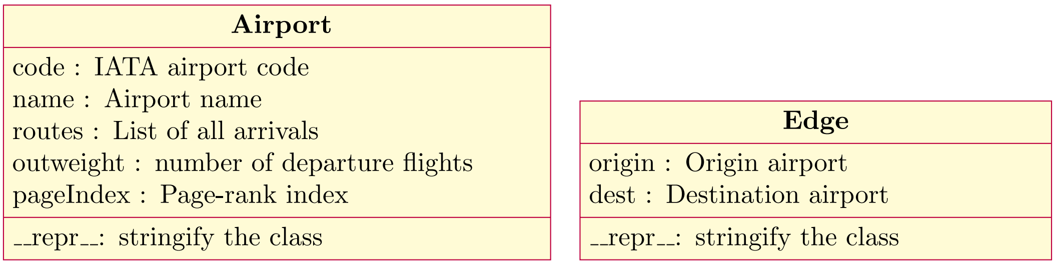

Airport
code : IATA airport code
name : Airport name
routes : List of all arrivals
outweight : number of departure flights
pageIndex : Page-rank index
__repr__: stringify the class
Edge
origin : Origin airport
dest : Destination airport
__repr__: stringify the class

Figure 1: Class diagrams

2.2 Difficulties and Choices

Laura!! I tried here to answer this question from the lab: Modify the pseudocode above to add the effect of these (virtual) edges efficiently: how many of them are there, and how much pagerank do they add in total to each vertex in particular?

Our main difficulty was dealing with the *airports* without departures(formally, *dangling nodes*). There are three accepted approaches for treating pages with no outgoing links[1]:

1. Eliminate such pages from the graph (iteratively prune the graph until reaching a steady state).

[1] More details about the data structure can be found in the lab: Session03

[2] In fact, 64MB it is not too big, It might be better to save the data as matrix in order to save data manipulation.

2. Consider such pages to link back to the pages that link to them.
3. Consider such pages to link to all web pages (effectively making an exit out of them equivalent to a random jump).

The first approach can lead to loss of information. consider the graph in figure[2] that represent our network, we can see that we can lose information about airports. The second and third approaches are more robust and promise that we will not lose information, we decided to add link to all nodes (TRUE??). However, we don't keep it on the data structure itself, we add it to the iterative computation of the page-rank, therefore we won't "pay" for n^2 new edges.

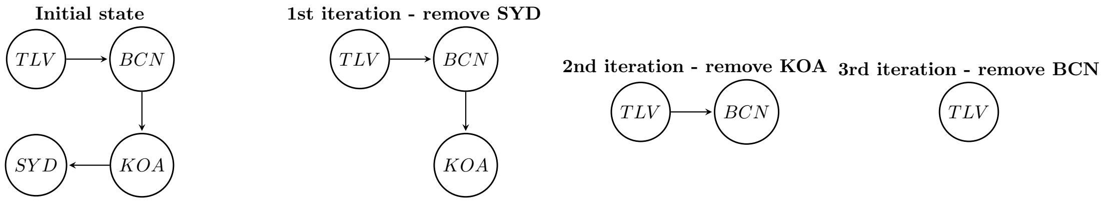

Initial state
TLV
BCN
SYD
KOA
1st iteration - remove SYD
TLV
BCN
KOA
2nd iteration - remove KOA
TLV
BCN
3rd iteration - remove BCN
TLV

Figure 2: 1st approach

3 Experiments and Observations

different dumping factors
about the converges parameter
other?

References

[1] Yahoo Labs Ronny Lempel. Introduction to search engine technology. https://webcourse.cs.technion.ac.il/236375/Winter2013-2014/ho/WCFiles/lec2-linkAnalysisIntro.pdf, Winter 2013-2014.