Draft version April 9, 2020
Typeset using LaTeX default style in AASTeX62

IN4200: Home exam I: Analysing web graphs.

Kandidatnr

1. INTRODUCTION

In this project we are going to implement a function for analysing data froma web graph file. The goal of this project is to outline the algorithms required for reading such a file and organising the data in both a 2D table and on a compressed row storage format. Using both of these formats for storing the data we will show algorithms for computing the number of times each web page participates in a mutual linkage with another web page. The final goal of this project is to rank the webpages in a web graph with respect to the number of mutual linkages every web page participates in. The algorithms were tested with and without parallelization using openMP.

2. METHOD

2.1. *Format of a web graph file.*

When storing the data from the web graph file, one has to take into account how a web graph file stores its information. An example of a web graph file is shown below.

```
# Directed graph (each unordered pair of nodes is saved once): 8-webpages.txt
# Just an example
# Nodes: 8 Edges: 17
# FromNodeId     # ToNodeId
0    1
0    2
1    3
2    4
2    1
3    4
3    5
3    1
4    6
4    7
4    5
6    0
5    7
6    4
6    7
7    5
7    6
```

Nodes in this file represent the number of web pages that are included in the web graph. All nodes are given an Id with integer numbers ranging from zero to the amount of Nodes in the web graph. Edges represent the amount of times there is a link from one web page to another. The two columns FromNodeId and ToNodeId is the main data we are analysing. FromNodeId represents the Id of the web page with a link to the corresponding ToNodeId. From the first row in the example file we can see an example of this where Node 0 has a link to Node 1 and so forth.

2.2. *Storing the data from the file*

When storing the file in a 2D table format one has to allocate a matrix with dimension Nodes$\times$Nodes. Along each row of the matrix we will store the amount of web pages that are linked to the given web page designated by the row,

i.e, the first row designates the first web page and each element on that row designates designates all the other web pages in the web graph file. If a web page has a link to the web page belonging to the current row, the index of the the web page along the row will be given a value of 1. If it does not link to the web page belonging to the row it will be given a value of zero. This continues for every row untill all web pages in the file has its own designated row thereby. To store the data in a $2D$ table format, we allocate a 2D-array and set all elements to zero. When reading through the web graph file we set the web pages linking to the web page belonging to the designated row as `table2D[to][from] = 1`, where `to` is the current value on the ToNodeId column and `from` is the current value on the FromNodeId column. When doing this we have to make sure that we do not count if `from=to` as one web page can not link to itself giving the 2D table zeroes on the diagonal.

When working with large web graph files, the 2D table format will store a lot of unneccesary information as zeroes. It is therefore convenient to introduce a much more, allthough a little bit more complicated, convenient format in terms of RAM usage. In this project we use the so called crompressed row storage format (CRS). In stead of storing all values as a matrix where alot of storage will be used to represent zeroes, we will now store our data in two arrays `row_ptr` and `col_idx`. The `col_idx` array includes the index for each row where we have a value of one in the 2D table. This array is of length N_{links} which corresponds to the number of Edges in the web graph file. The `row_ptr` array, starting at zero designating the first row, includes the index at which the `col_idx` array changes row. This array is of length $N+1$ where N corresponds to the number of Nodes in the web graph file. With these two arrays we can store the same information as in the 2D table but freeing up a lot of memory as we are only storing $N+1+N_{edges}$ integers instead of N^2 integers to account for all the zeroes in the 2D table. When creating these two arrays we start by reading through the web graph file and storing all the FromNodeId's and ToNodeId's in two arrays. At the same time we update the `row_ptr` array. Every time there is a linkage, it means that there should be one more element on each row belonging to the given web page being linked to. This is is done by adding one to the array as `row_ptr[to + 1]++`, where we index at `[to + 1]` since index zero is set to zero. When