In this project, you will use OpenMP to parallelize Google’s ranking process and apply it on the Erdos Web Graph which can be downloaded at http://web-graph.org/. The ranking will be done by carrying out the following iteration:

\[
\begin{bmatrix}
1 \\
1 \\
\vdots \\
1
\end{bmatrix}
\]

\[
\begin{array}{l}
\text{Repeat} \\
r^{(t+1)} = \alpha P r^{(t)} + (1 - \alpha) c \\
\text{until} \quad ||r^{(t+1)} - r^{(t)}||_1 \leq \varepsilon
\end{array}
\]

Here

- \( c = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \)
- Take \( \alpha \) as 0.2
- \( ||r^{(t+1)} - r^{(t)}||_1 = \sum |r_i^{(t+1)} - r_i^{(t)}| \)
- \( \varepsilon \) is a small number, e.g. \( 10^{-5} \)

The matrix \( P \) is to be stored in CSR format. CSR format is explained below. You should provide a write-up of how you implemented your project and the following results:

a) The timings obtained:

<table>
<thead>
<tr>
<th>Test No.</th>
<th>Scheduling Method</th>
<th>Chunk Size</th>
<th>No. of Iterations</th>
<th>Timings in secs for each number of threads</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) The first 5 hosts that have the highest rankings.

**Google Ranking Process**

You can take a look at the following to learn about the Google ranking process.

- http://infolab.stanford.edu/~backrub/google.html
CSR Matrix Storage Format
Consider the following sparse matrix storage scheme, called compressed sparse row (CSR) format. An example of a matrix represented in this format is given below:

\[
P = \begin{bmatrix}
11 & 0 & 13 & 14 & 0 \\
0 & 0 & 23 & 24 & 0 \\
31 & 32 & 33 & 34 & 0 \\
0 & 42 & 0 & 44 & 0 \\
51 & 52 & 0 & 0 & 55
\end{bmatrix}
\]

The above matrix will be stored as follows:

row_begin = [ 0  3  5  9 11 14 ]
values = [ 11 13 14 23 24 31 32 33 34 42 44 51 52 55 ]
col_indices = [  0   2   3   2   3   0   1   2   3   1   3   0   1   4 ]

Let \( N \) stand for the number of non-zero entries in the matrix and \( n \) stand for the number of rows. The array values contains non-zero entries in the matrix in row wise order. The array col_indices gives the corresponding column indices of these values. The array row_begin of size \( n+1 \) stores the beginning index of each row in the values (and col_indices arrays). The last entry in row_begin stores \( N+1 \) so that the expression \( \text{row_begin}[i+1]-\text{row_begin}[i] \) gives the number of nonzeros in row \( i \).