CMPE 478: Parallel Processing
Fall 2013, Homework 2
(This project can be done in groups of 2 students)

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:

$$r^{(0)} = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}$$

Repeat
$r^{(t+1)} = \alpha\, P r^{(t)} + (1 - \alpha)\, c$
until $||r^{(t+1)} - r^{(t)}||_1 \leq \varepsilon$

Here

- $c = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}$
- Take α as *0.2*
- $||r^{(t+1)} - r^{(t)}||_1 = \sum | r_i^{(t+1)} - r_i^{(t)} |$
- ε 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:

Test No.	Scheduling Method	Chunk Size	No. of Iterations	Timings in secs for each number of threads							
				1	2	3	4	5	6	7	8
1											
2											
…											

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://www.cmpe.boun.edu.tr/~ozturan/etm555/google.pdf
- http://infolab.stanford.edu/~backrub/google.html
- http://www.rose-hulman.edu/~bryan/googleFinalVersionFixed.pdf

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 nonzero 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 **`row_begin[i+1]-row_begin[i]`** gives the number of nonzeros in row i.