# MPI PageRank

## 1 Introduction

**PageRank** [1] is an algorithm for ranking webpages based on hyperlink structure. A vertex $v$'s PageRank score is a measure of its importance in a network, and is based on the importance of vertices which have edges pointing to $v$. The algorithm that you will implement goes as follows:

**Algorithm 1** PageRank

```
1: Input: Directed graph $G = (V, E)$, damping factor $d$, and tolerance $\epsilon$.
2: Output: PR scores of each $v \in V$.
3: $PR_0(v) = 1/|V|; \forall v \in V$
4: repeat
5:     $PR_t(v) = \frac{1-d}{|V|} + d \sum_{(i,v) \in E} \frac{PR_{t-1}(i)}{|(i,*) \in E|}$
6:     $r = ||PR_t - PR_{t-1}||_F$
7: until $r < \epsilon$ or $t >$ maximum iterations.
```

## 2 Assignment

Design and implement a distributed-memory parallel algorithm for computing PageRank. Your algorithm should use a 1D distribution of the graph (i.e., each vertex and its PageRank score belongs to exactly one MPI rank).

Your algorithm **must** be memory scalable. This means that no rank can allocate an array of size $|V|$ or $|E|$, and should only store its own vertices and PageRank scores, and those of non-local vertices connected by incoming edges. Any accesses to PageRank values which are non-local must be performed in constant time during the computation (not counting communication costs).

Do not attempt to read the input files with multiple MPI ranks at once. Use the root MPI rank to read in chunks of the input graph and send communicate to the remaining ranks.

### 2.1 File format

We store our directed, unweighted, sparse graphs in a file format which encodes the adjacency list of each vertex. A graph with $V$ and $E$ edges is encoded with a file containing $V+1$ lines. The first is a header with two space-separated integers $V$ and $E$. The following $V$ lines encode the adjacency list of each vertex. Line $v$ has space-separated integers which encode the *outgoing* edges from the vertex $v-1$ ($-1$ due to the header occupying a line). For example, Figure 1 is represented with the following file:

```
4 5
2
3
4
1 2
```

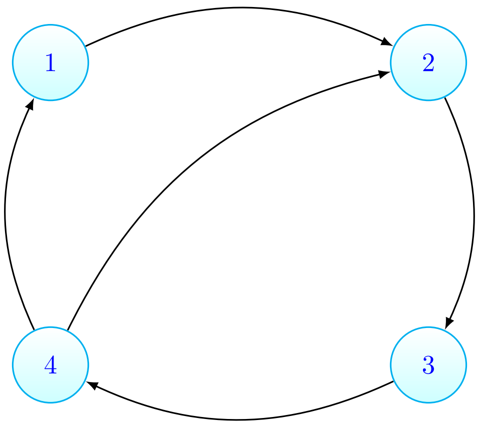

Figure 1: A simple graph.

### 2.2 Deliverables

We have provided you with a serial code for computing PageRank in C. You should modify the provided code and parallelize it using MPI.

Your program should be executed via:

```
$ mpirun -np 16 ./pagerank A.graph A.pr.txt
```

As always, your submission must include a writeup explaining your parallel algorithm and implementation details. Your writeup should include a complexity analysis of your algorithm. What are the parallel overheads? Is your algorithm ideally suited for some types of graphs but not others?

Also provide timings on `A.graph`, `B.graph`, and `live-journal.graph` using 1, 2, 4, 8, 16, and 32 MPI ranks. You should either use MPI's or the timing utilities used in previous assignments, and provide the average time per iteration. Use this formatting:

```
Number of iterations: 10 average time: 0.138s
```

Notice that all three datasets have very similar sizes. How does your performance differ across the datasets? If there is variation, why?

## References

[1] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.