PES UNIVERSITY ONLINE

BIG DATA

Big Data Algorithms
Page Rank Computations

K V Subramaniam
Computer Science and Engineering

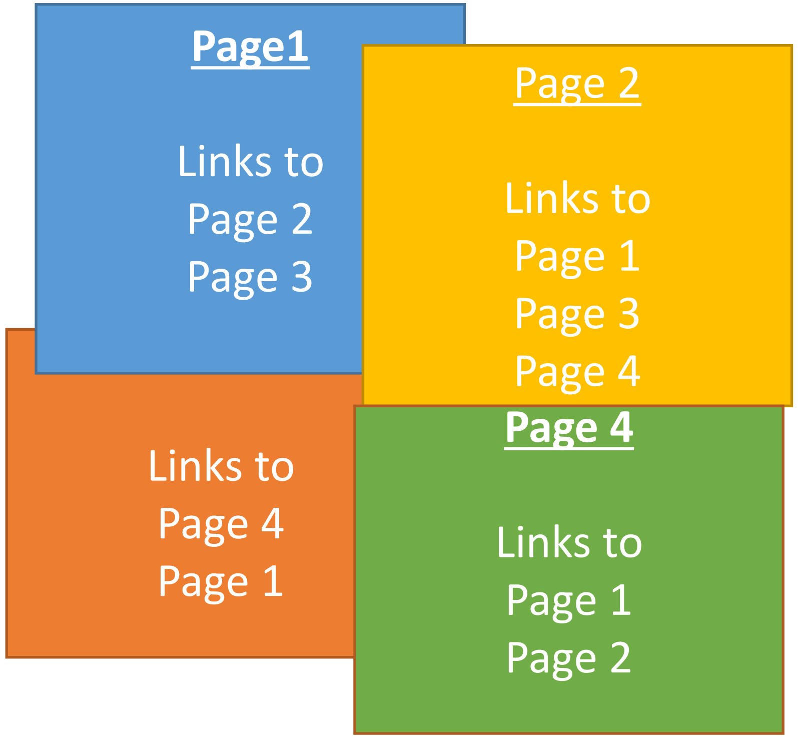
Matrix Multiplication

Overview

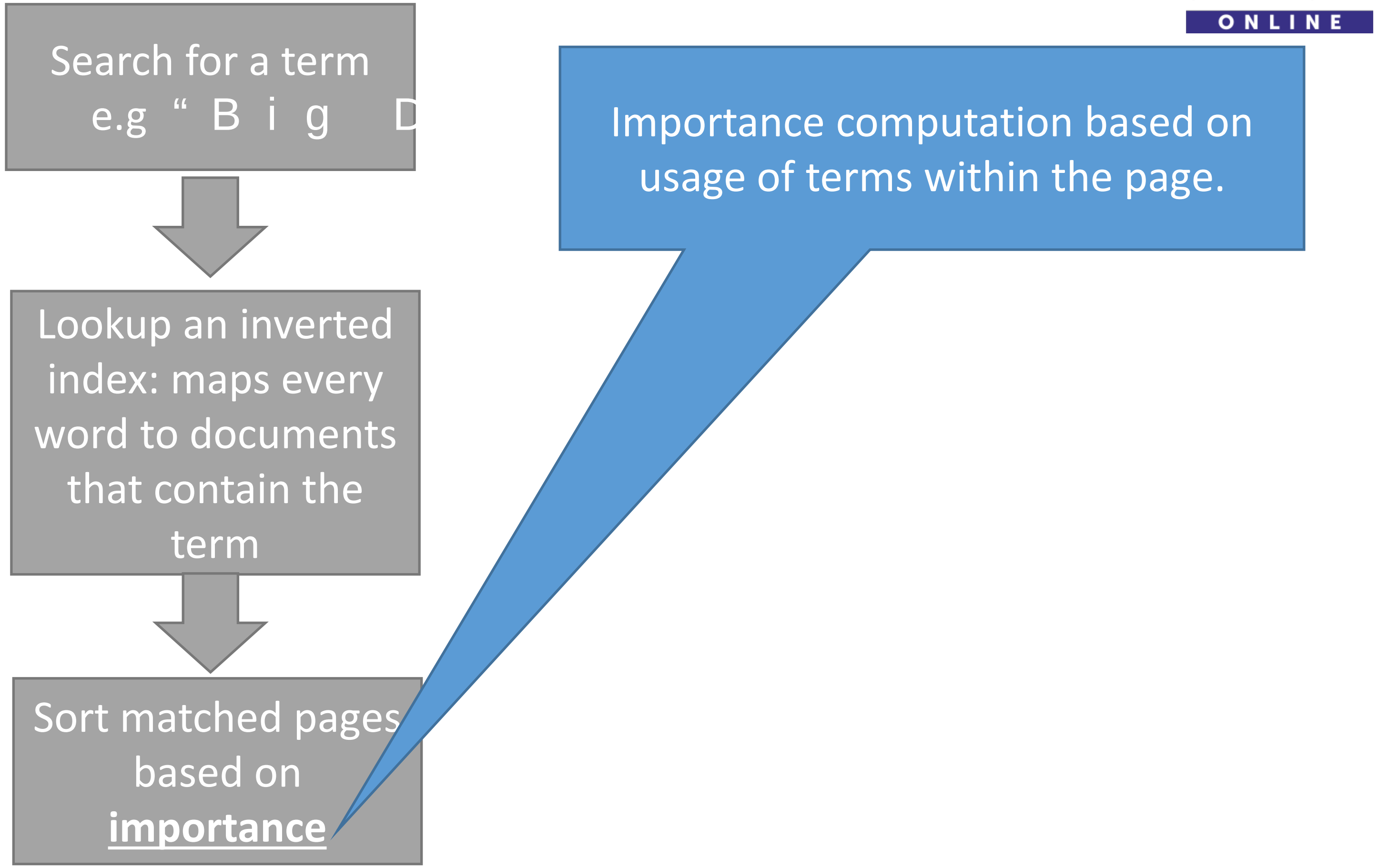
- Page Rank
- Source
  - Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.
  - http://infolab.stanford.edu/~ullman/mmds/book.pdf

Page Rank : Overview

Search Engine working



Pages in the WWW



Search for a term e.g “Big D

Lookup an inverted index: maps every word to documents that contain the term

Sort matched pages based on **importance**

Importance computation based on usage of terms within the page.

Issues with early search engines

- Term Spam
  - Unethical use add terms multiple times

- Example
  - I could setup a page with PES University occurring a 1000 times in the page.
  - Search Engine would think that my page is an authority for PES Univesity mark it as ***<u>important</u>***
  - How to fix this?

One possible solution

- Instead of taking terms in my page
- Consider how many pages are pointing into my page?
- Will this not solve the problem.

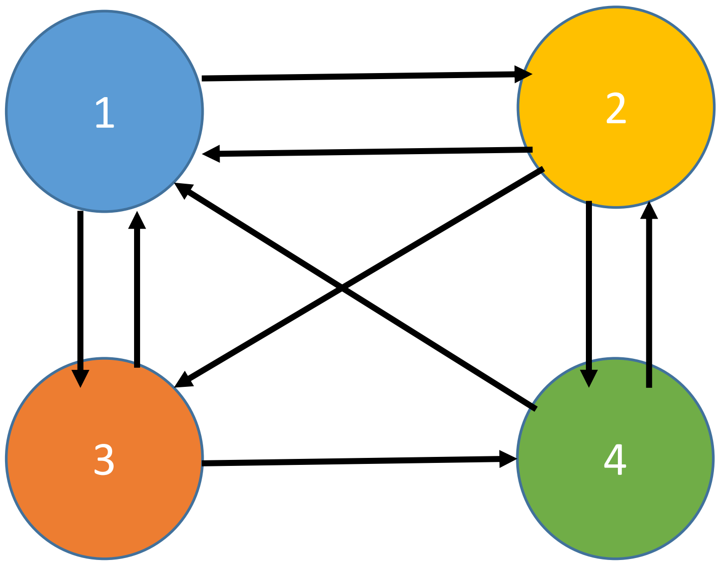
- NO!!
- Can add many spam pages that point to my page.

How to solve this

- Two techniques
  - Consider random surfers starting at a random pages
    - What fraction of surfers end up at my page?
    - This gives an indication of the importance of my page.
    - *Page Rank*
  - Take into account the terms near the links present in pages that point to my page
    - Gives an indication of how relevant my page is.

Page Rank : Transition Matrix

Adjacency Matrix

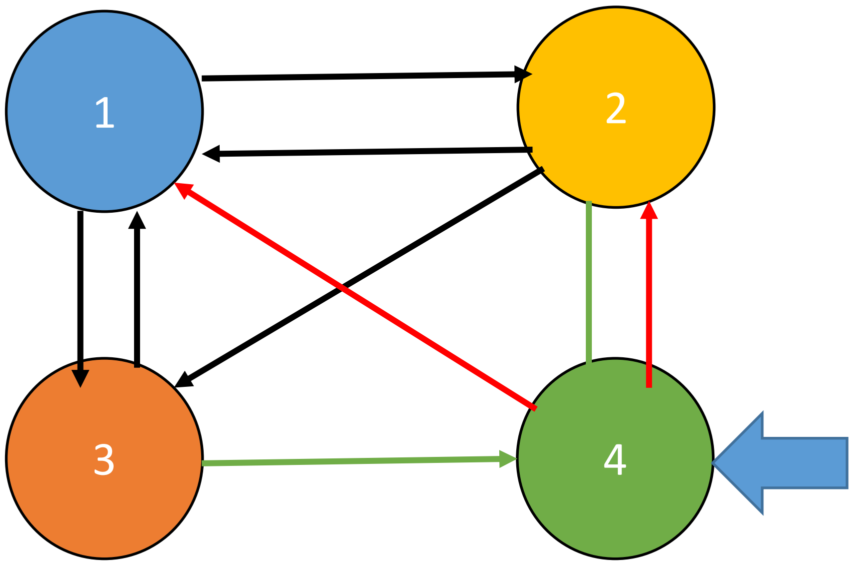


$$\begin{matrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{matrix}$$

Let us start from our graph model of the internet

- Adjacency matrix represents which pages are reachable from a given page

Adjacency Matrix



If we are at node 4 (page 4) and we follow a link out

Then

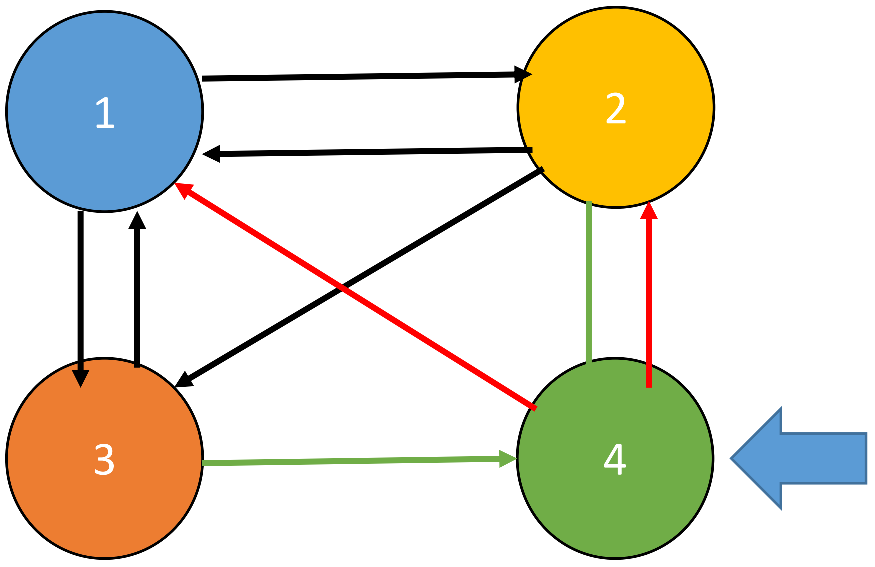
- the pages we can visit are 1 and 2
- 3 is not reachable directly from 4

To summarize for node 4

#in links = 2

#out links = 2

Random surfer



Consider, a random surfer starting from page 4

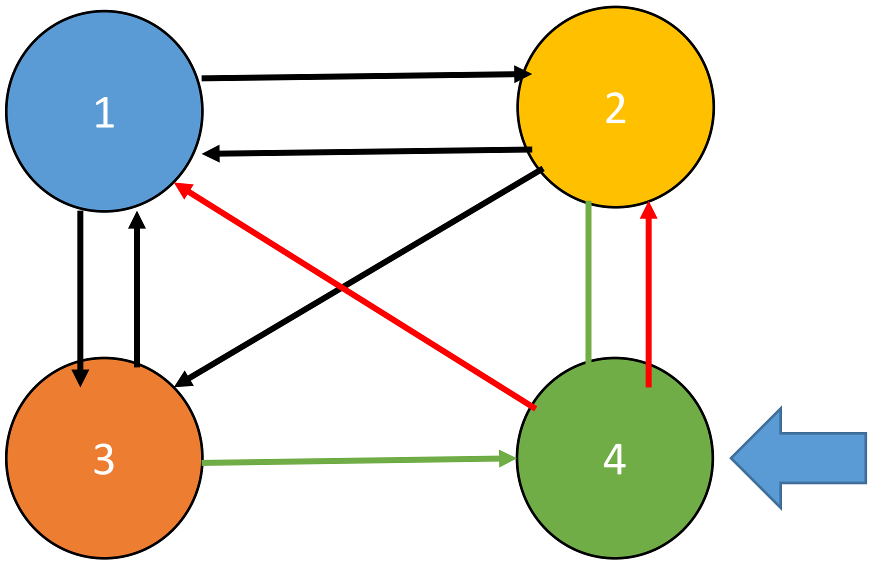
Assume, that we have equal probability of taking either out link

So,

P(transition at node 4) = 1/#outlinks

= ½

Transitioning at a node

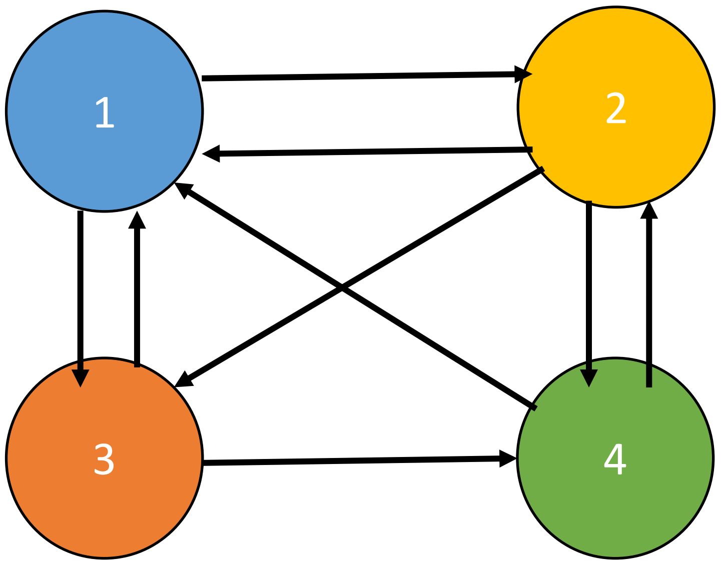


For node for we can represent the probabilities of directly transitioning

- To every other page as a column vector

$$\begin{array}{cc} & \text{Source} \\ \text{Dest} & \begin{array}{c} 1/2 \\ 1/2 \\ 0 \\ 0 \end{array} \end{array}$$

Transition Matrix



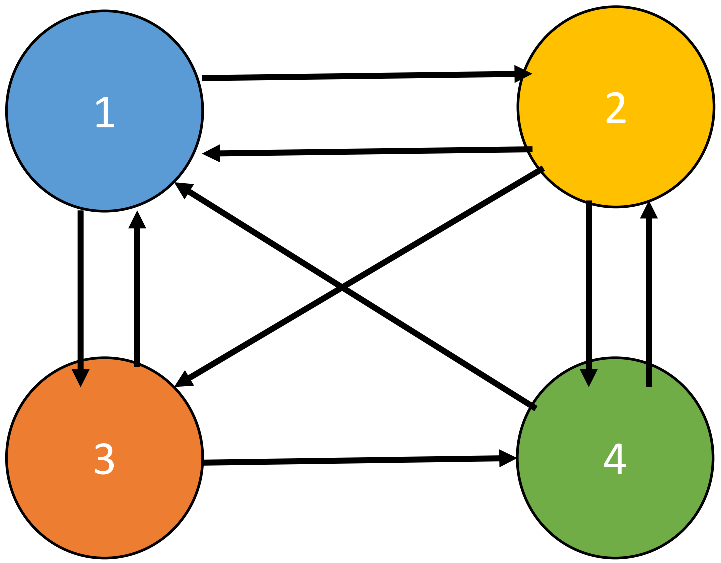
For the entire graph with n nodes, this would be a *nxn* matrix

- Transition matrix **(M)**
- Each entry represents the probability of transition from a source to a destination
- Source column #
- Destination row #

$$\begin{array}{c|cccc} & \text{Source} \rightarrow & & & \\ \text{Dest} \downarrow & 0 & 1/3 & 1/2 & 1/2 \\ & 1/2 & 0 & 0 & 1/2 \\ & 1/2 & 1/3 & 0 & 0 \\ & 0 & 1/3 & 1/2 & 0 \end{array}$$

Page Rank : Following the random surfer

Random surfer - initialization



The random surfer can start off on any of the nodes

So each node has equal chance of being a starting node.

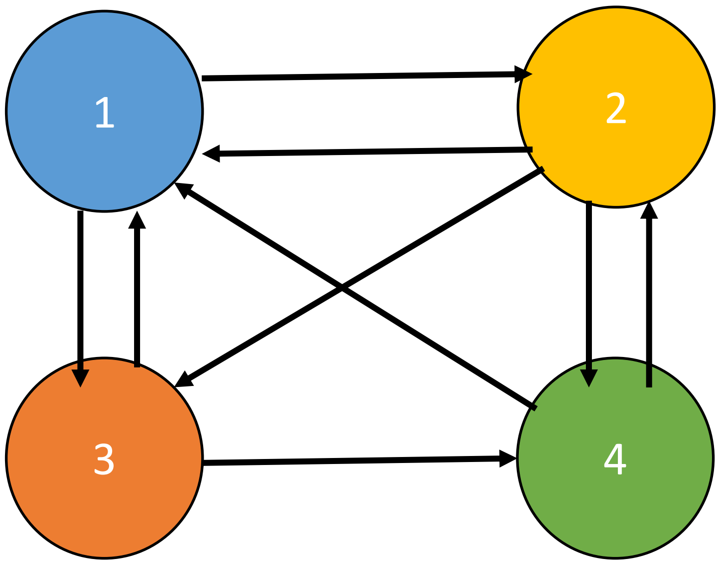
Lets call the relative importance of each node - ***importance*** represented as vector ***v***

Our first guess at this is called $\boldsymbol{v_0}$

$$\boldsymbol{v_0} = \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{pmatrix}$$

Node importance

Random surfer - movement



As random surfer moves once

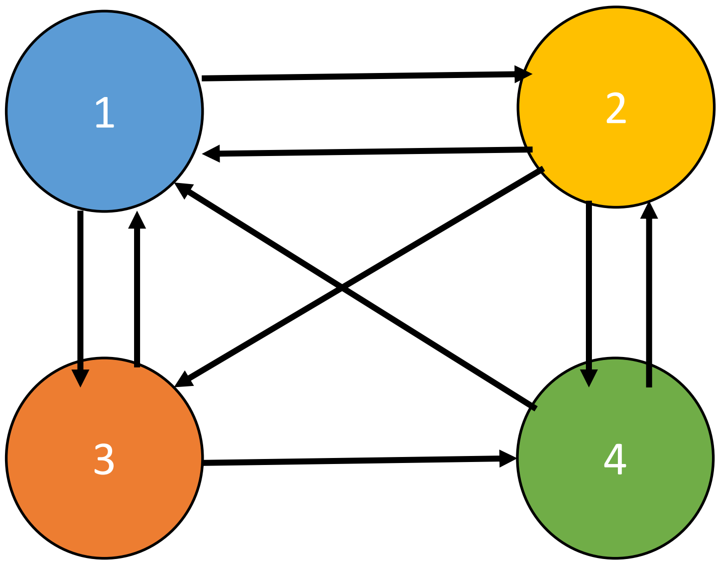
Compute chance they will land up in each of the nodes

For each node we compute the probability of ending up at that node based on previous node

Multiply transition matrix and ***v***

$$\begin{bmatrix} 0 & 1/3 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \\ 1/2 & 1/3 & 0 & 0 \\ 0 & 1/3 & 1/2 & 0 \end{bmatrix} \overset{\boldsymbol{v_0}}{\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}} = \begin{bmatrix} 1/3*1/4+1/2*1/4+1/2*1/4 \\ \text{½}*1/4+1/2*1/4 \\ \text{½}*1/4+1/3*1/4 \\ 1/3*1/4+1/2*1/4 \end{bmatrix} = \overset{\boldsymbol{v_1}}{\begin{bmatrix} 1/3 \\ 1/4 \\ 5/24 \\ 5/24 \end{bmatrix}}$$

Random surfer - movement



With each move of the random surfer, we will repeat the computation

Multiply transition matrix and ***v***

$$\begin{pmatrix} 0 & 1/3 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \\ 1/2 & 1/3 & 0 & 0 \\ 0 & 1/3 & 1/2 & 0 \end{pmatrix} \overset{\boldsymbol{v_1}}{\begin{pmatrix} 1/3 \\ 1/4 \\ 5/24 \\ 5/24 \end{pmatrix}} = \boldsymbol{v_2}$$

Page Rank : As an Eigen vector problem

- *x* satisfies the equation to the right.
  - Generally: some set of equations for *x.*
- How do we solve for *x*?
- Iterative algorithm: widely used in Big Data
  - *i* = number of times we have looped,
  - x_i = value of *x* on i^{th} iteration,
  - Calculate some “error term” based up x_i.
    - Shows how far x_i is from the correct value.
    - E.g., $Ax_i - x_i$
  - Derive x_{i+1} based upon x_i and error term.
  - Loop over steps 3 and 4 until error term is small.
- Iterative algorithm calculates $x_0\, x_1\, x_2\, x_3$ … which eventually converges to a good (or the good) solution.

$$A \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Iterative Algorithms: Fixed Point Iteration

- Suppose the equation that x satisfies has the form x=f(x)
- x may be a matrix or vector, then the following simplified iterative algorithm frequently works

- Fixed point iteration
    - Initialize x_0 to some value
    - Calculate $x_{i+1} = f(x_i)$
    - Loop over step 2 until either
        - error term = $|x_{i+1}-x_i|$ is small i.e., not much change between x_{i+1} and x_i
        - or for some maximum number of iterations.

https://en.wikipedia.org/wiki/Fixed-point_iteration#:~:text=In%20numerical%20analysis%2C%20fixed%2Dpoint,the%20fixed%20point%20iteration%20is

Eigenvalues and eigenvectors

- Multiplying a *mxn* matrix by an *n* element vector gives an *m* element vector.
- If matrix is *nxn*, the product will also be an *n* element vector. $A\mathbf{v} = \lambda\mathbf{v}$
- Suppose
  - Multiplying ***A*** by ***v*** gives back the same vector apart from a scale change .
  - λ is called the eigenvalue and ***v*** the eigenvector.
  - Many problems (e.g., page rank) can be converted into finding eigenvalues of a matrix.

Page Rank as Eigenvector Problem –Eigenvector Computation

- Initialize ***v*** to be a vector of equal values.

- Loop
  - $\boldsymbol{v}_{i+1} = \boldsymbol{A}\boldsymbol{v}_i$

- Until there is little difference between $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$
- At this point
  - $V = \boldsymbol{A}\boldsymbol{v}$

- ***v*** is the eigenvector and page rank of the transition matrix of the web graph.

Page Rank as Eigenvector Problem –Overview

- We want to compute the *importance* (page rank) of each Web page.

- Assume that
  - If a page has importance *I*.
  - *n* links to other pages
  - It distributes it's importance *I* among the *n* links equally (*I*/*n*)
- Use this assumption to calculate page rank

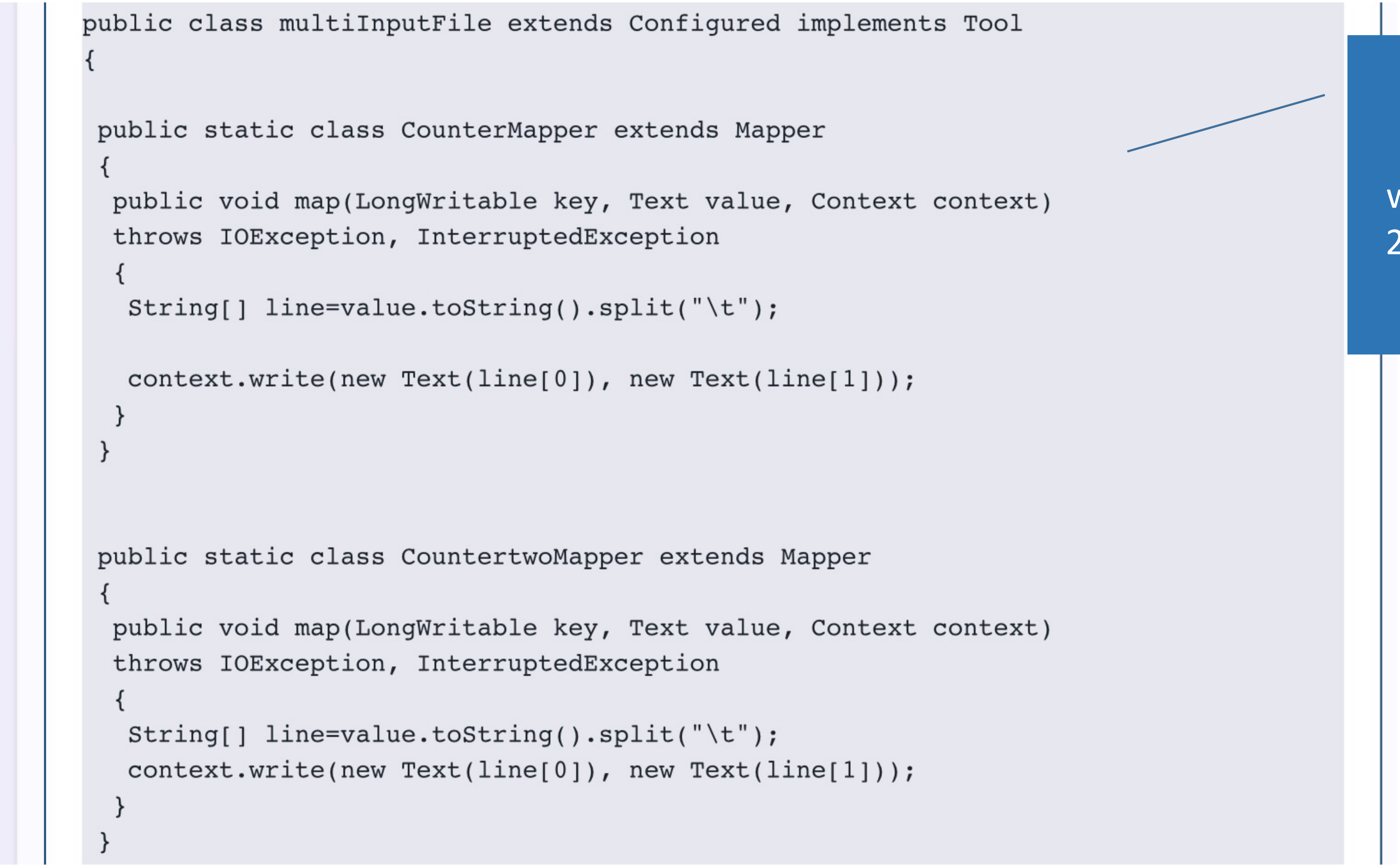
Page Rank : Map Reduce Implementation

Implementation in Map Reduce

- We need support for
  - Handling multiple files as input to the mapper
    - The initial page rank
    - Part of the Transition Matrix
  - Iteration over multiple matrix-vector multiplication rounds

BIG DATA

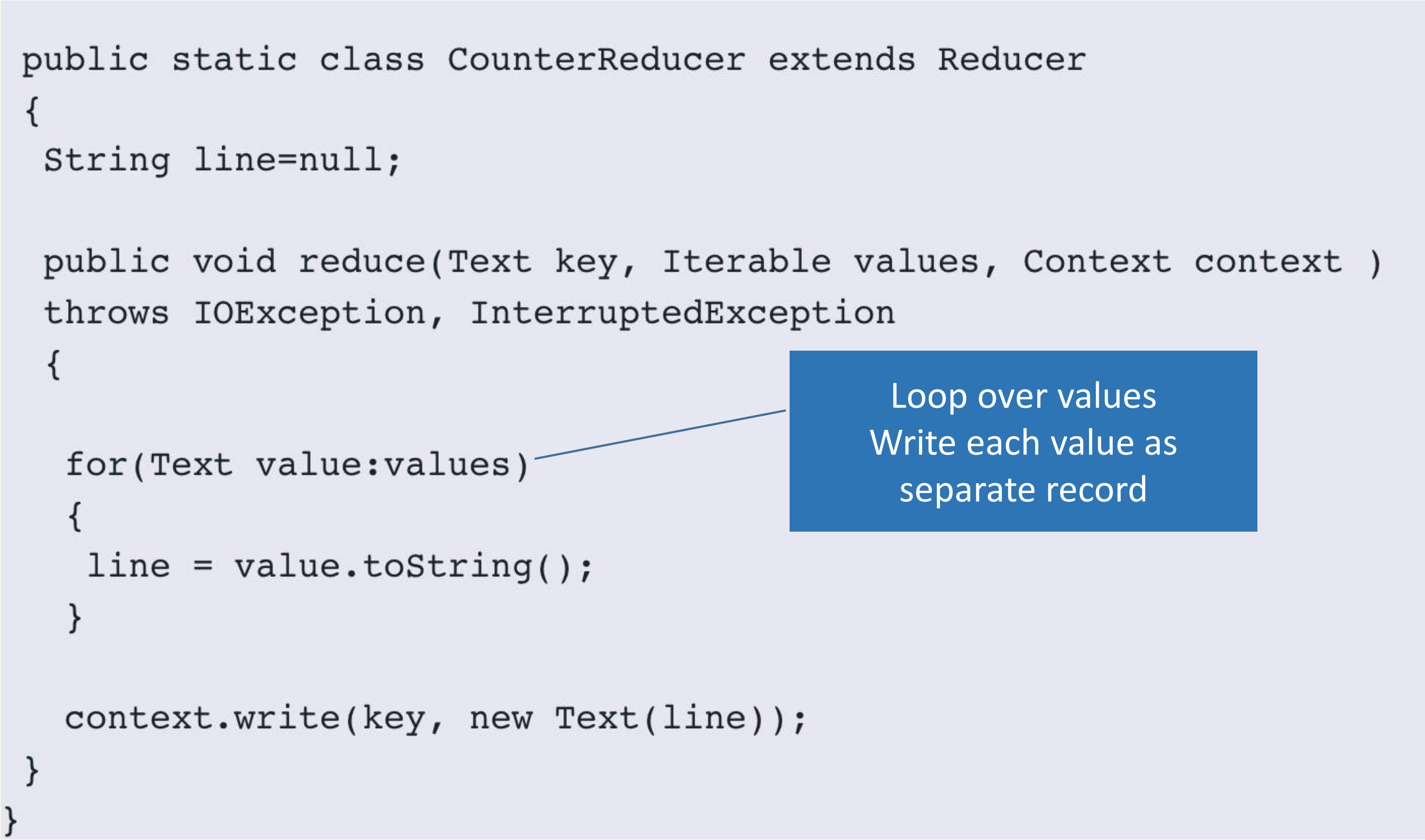
Hadoop Multiple Input Files –mapper



```
public class multiInputFile extends Configured implements Tool
{

 public static class CounterMapper extends Mapper
 {
  public void map(LongWritable key, Text value, Context context)
  throws IOException, InterruptedException
  {
   String[] line=value.toString().split("\t");

   context.write(new Text(line[0]), new Text(line[1]));
  }
 }


 public static class CountertwoMapper extends Mapper
 {
  public void map(LongWritable key, Text value, Context context)
  throws IOException, InterruptedException
  {
   String[] line=value.toString().split("\t");
   context.write(new Text(line[0]), new Text(line[1]));
  }
 }
```

Mappers write first word as key, 2nd word as value

Hadoop Multiple Input Files –reducer

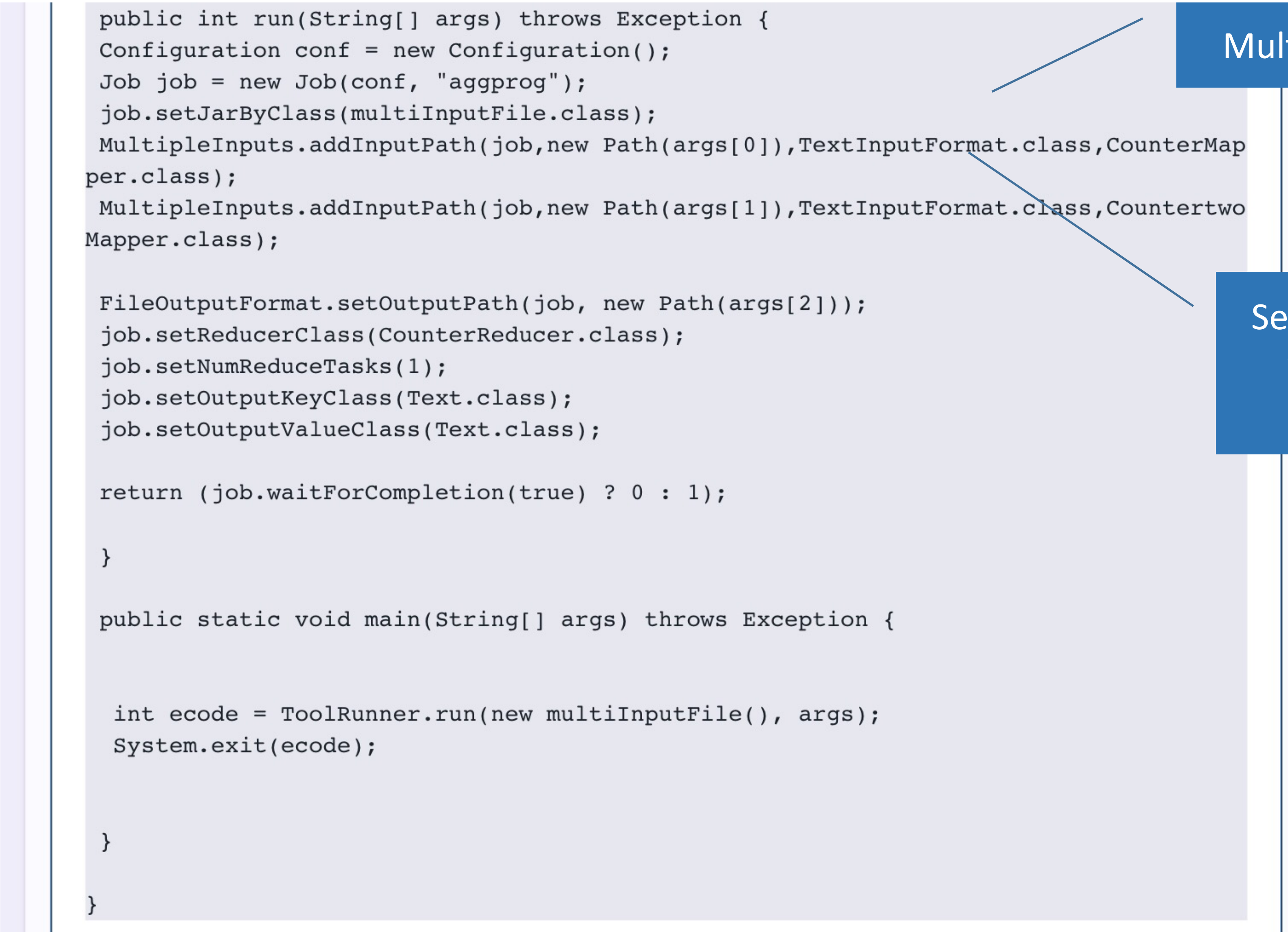


```
public static class CounterReducer extends Reducer
{
 String line=null;

 public void reduce(Text key, Iterable values, Context context )
 throws IOException, InterruptedException
 {

  for(Text value:values)
  {
   line = value.toString();
  }

  context.write(key, new Text(line));
 }
}
```

Loop over values
Write each value as
separate record

BIG DATA

Hadoop Multiple Input Files –job



```
 public int run(String[] args) throws Exception {
 Configuration conf = new Configuration();
 Job job = new Job(conf, "aggprog");
 job.setJarByClass(multiInputFile.class);
 MultipleInputs.addInputPath(job,new Path(args[0]),TextInputFormat.class,CounterMap
per.class);
 MultipleInputs.addInputPath(job,new Path(args[1]),TextInputFormat.class,Countertwo
Mapper.class);

 FileOutputFormat.setOutputPath(job, new Path(args[2]));
 job.setReducerClass(CounterReducer.class);
 job.setNumReduceTasks(1);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(Text.class);

 return (job.waitForCompletion(true) ? 0 : 1);

 }

 public static void main(String[] args) throws Exception {


  int ecode = ToolRunner.run(new multiInputFile(), args);
  System.exit(ecode);


 }

}
```

Multiple input class

Set input file for each mapper

Iteration

- One MR step produces a estimate of ***v***
  - ***The file is stored <u>in HDFS</u>***

- At the end of the iteration, we need to compare this with the previous iteration estimate of ***v***

- After that, we use this file as the input for the next step of MR.

THANK YOU

K V Subramaniam, Usha Devi

Dept. of Computer Science and Engineering

subramaniamkv@pes.edu

ushadevibg@pes.edu