**Matrix-Vector Multiplication**

Matrix *M* dimension *n x n*

*mij* row *i* and column *j*

Vector *v* length *n*

*vj* denotes *jth* element  
Matrix-vector product is a vector *x* of length *n*

xi =Σ*mij vj*

**Matrix Multiplication**

Matrix *M* with element *mij* , Matrix *N* with element *njk*

Product P : *pik* = Σ*j mij njk*

Relation M (I, J, V) with tuples (i, j, *mij*)

Relation N (J, K, W) with tuples (j, k, *njk*) Sparse matrix, omit tuples for zero elements

**Distance Measures**

▪Jaccard similarity of two sets is the size of their intersection divided by the size of their union:

*sim* (C1, C2) = |C1 ∩ C2| / |C1 ∪ C2|

▪Jaccard distance:

*d* (C1, C2) = 1 - |C1 ∩ C2| / |C1 ∪ C2|

**Find Similar Documents**

1. Shingling-Convert documents to sets  
2. Min-Hashing-Convert large sets to short

signatures, while preserving similarity

3. Locality- Sensitive Hashing-Focus on pairs of signatures likely to be from similar documents

▪Candidate pairs!

Important: Similarities of signatures and similarities of shingles MUST BE related

Not every hashing function is applicable Need one that satisfies the following:

▪if *sim*(D1,D2) is high, then with high prob. *h*(D1) = *h*(D2)

▪if *sim*(D1,D2) is low, then with high prob. *h*(D1) ≠ *h*(D2)

Master node fail - Restart entire MapReduce job

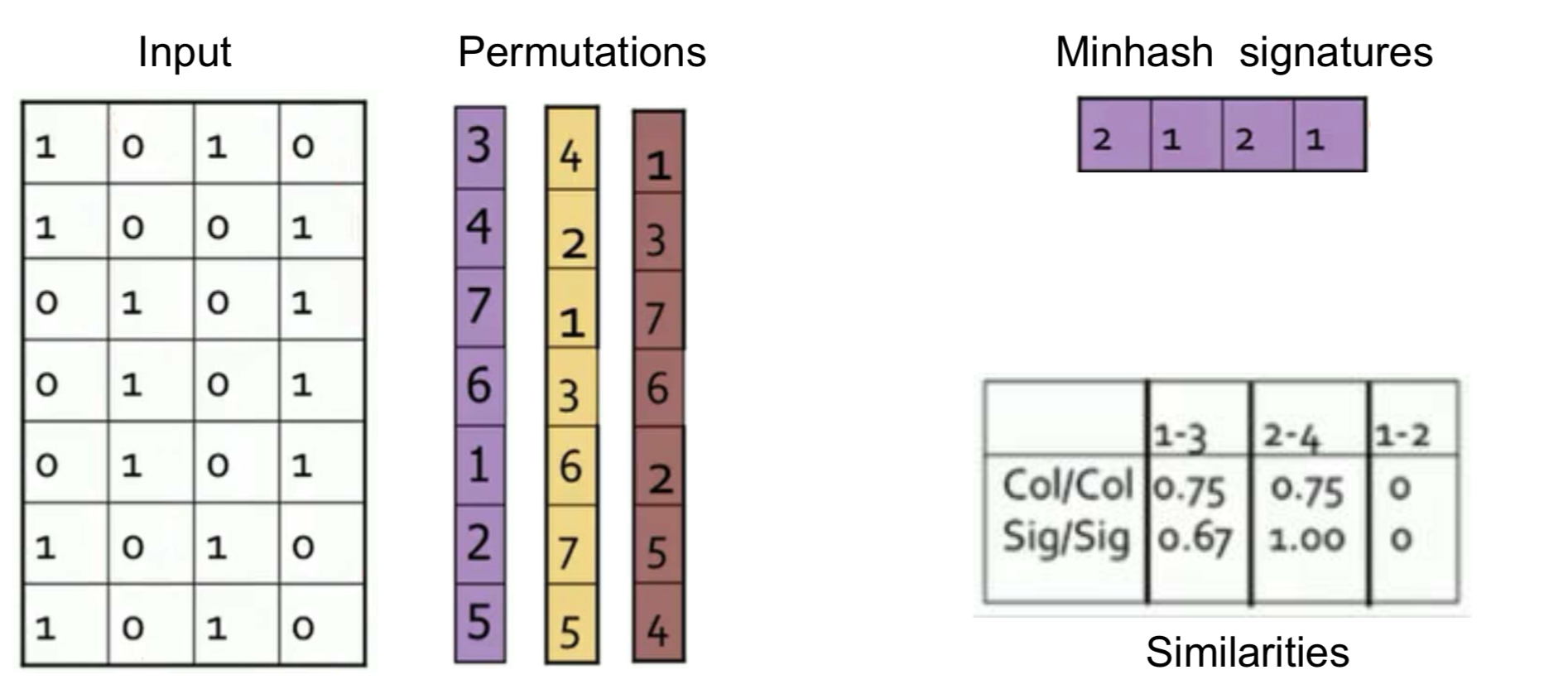
Compute node of a Map worker fail

▪Reset completed or in-progress map tasks at worker to idle

▪Restart all the map tasks assigned to this node

▪ Inform Reduce workers when task is rescheduled on another worker, location of input from that map task has changed

Compute node of a Reduce worker fail  
Only reset in-progress tasks to idle  
Restart these reduce tasks at another node



此处是指1所标注的那一行会成为第一行。

**Analysis of LSH**

▪*b* bands, *r* rows per band  
▪Columns C1 and C2 have Jaccard similarity *s*

▪Prob. that they have same minhash value

▪Pick any band (*r* rows)

▪Prob. that all rows in band equal = *s r*

▪Prob. that some row in band unequal = 1 – *s r* ▪Prob. that no band identical = (1 – *s r*)*b*

▪Prob. that at least 1 band identical = 1 – (1 – *s r*)*b*

**Association Rules**   
▪*{ i1, i2,...,ik } → j* means: “if a basket contains all of *i1,...,ik* then it is *likely* to contain *j*”

Rule evaluation metrics:

▪Support of a rule is the fraction of baskets that contain the itemset *{ i1,...,ik, j }*

▪Confidence of this association rule is the probability of *j* given *I* = {*i*1,...,*ik*}

conf(*I* → *j*) =support(*I U j)*/support(*I)*

*Measures how often item j occurs in baskets that contain I*

**Mining Association Rules**

▪Step 1: Find all frequent itemsets *I*▪Generate all itemsets whose support ≥ minsup

▪Step 2: Generate rules  
▪For every subset *X* of *I*, generate *X* → *I* − *X*

***Rule Genaration***

If I = {A,B,C,D}, then  
conf (ABC → D) ≥ conf (AB → CD) ≥ conf (A → BCD)

**Apriori Algorithm**

▪Two-pass approach reduce the need for main memory

▪Key idea: Apriori Principle

▪If an itemset is frequent, then all its subsets must be frequent

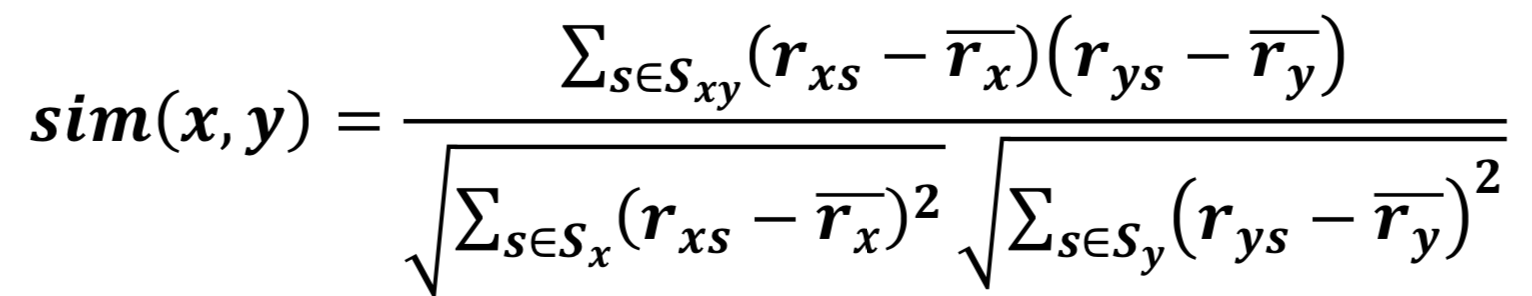
▪Apriori principle holds due to the anti-monotone property of support

∀*X*,*Y* :(*X* ⊆*Y*)⇒support(*X*)≥support(*Y*)

▪If item *i* does not appear in *s* baskets, then no superset containing *i* can appear in *s* baskets

**Pearson correlation coefficient**

*S* =items rated by both users *x* and *y*



**Distance measure**

1. Non-negativity: d(x, y) ≥ 0
2. Identity: d(x, y) = 0 if and only if x = y
3. Symmetry: d(x, y) = d(y, x)
4. Triangle Inequality: d(x, y) ≤ d(x, z) + d(z, y)

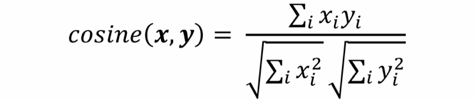
L2-norm = Euclidean Distance

d(x,y)=sqrt((x1 –y1)2 +(x2 –y2)2 +...+(xn –yn)2)

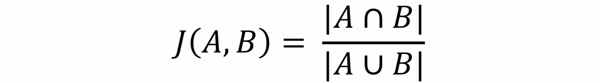
L1-norm = Manhattan Distance

d(x,y) = |x1 – y1| + |x2 – y2| + ...+ |xn – yn|

Cosine Distance



Jaccard Distance



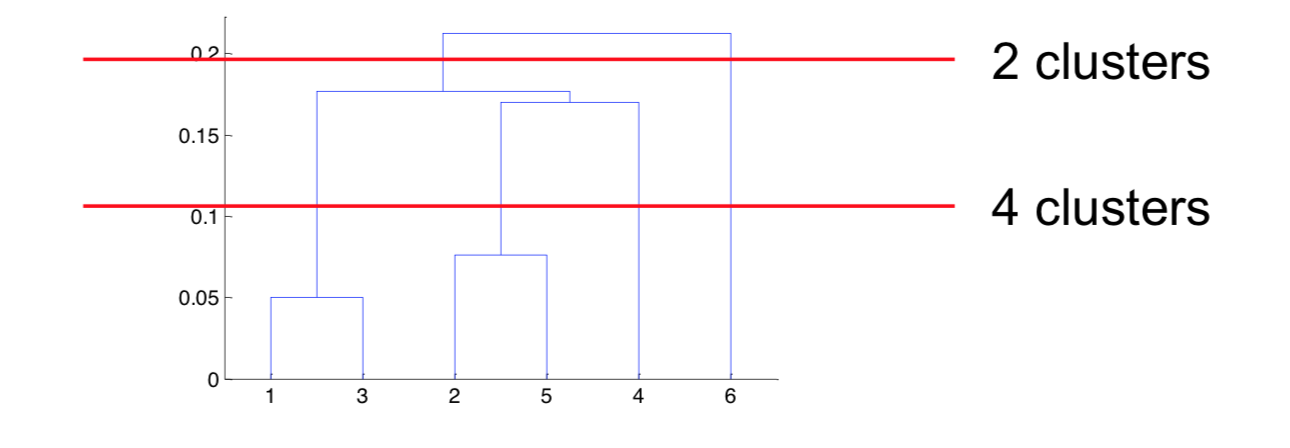
Edit Distance

Miniumum number of inserts and deletes of characters needed to convert one string into another

Clustering

**Hierarchical Clustering**

Repeatedly combine two nearest clusters



When to Terminate?

▪Pre-determined *number of clusters*▪When merging two clusters leads to a “bad” cluster

▪Diameter of merged cluster exceeds some threshold ▪Diameter exceeds average diameter by a wide margin  
▪Density of cluster falls below some threshold

*K*–Means Algorithm

**BFR Algorithm**

Keep track of 3 sets of points in memory

▪Discard set (DS):  
▪Points close enough to a centroid to be summarized

▪Compression set (CS):  
▪Groups of points that are close together but not close

to any existing centroid

▪These points are summarized, but not assigned to a cluster

▪Retained set (RS):  
▪Isolated points to be assigned to a compression set

For each cluster, discard set (DS) is summarized by

▪Number of points, *N*

▪ Vector *SUM* whose *i*th component is the sum of the coordinates of the points in the *i*th dimension

▪Vector *SUMSQ* where *i*th component = sum of squares of coordinates in *i*th dimension

▪2*d* + 1 values represent any size cluster (*d* = number of dimensions)

▪Centroid (Average in each dimension) is given by SUM*i* / *N*

▪SUM*i* = *i*th component of SUM  
▪Variance of a cluster’s DS in dimension *i* is given by

(SUMSQ*i* / *N*) – (SUM*i* / *N*)2  
▪Standard deviation is the square root of variance

Find those points that are “sufficiently close” to a cluster centroid and add those points to that cluster and the DS

▪These points are so close to the centroid that they can be summarized and then discarded

▪Use any main-memory clustering algorithm to cluster the remaining points and the old RS

▪Clusters go to the CS; outlying points to the RS

DS set: Adjust statistics of the clusters to account for the new points

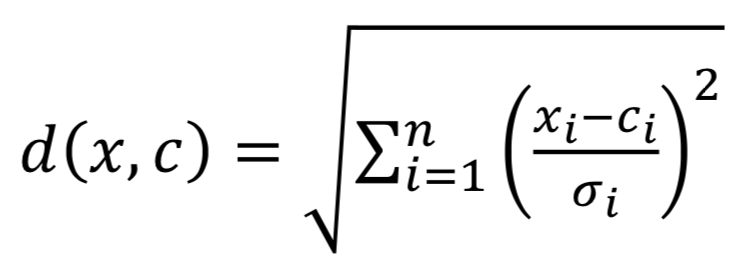
▪Add *N*s, *SUM*s, *SUMSQ*s

▪Consider merging compressed sets in the CS  
▪Add corresponding values of *N*s, *SUM*s, *SUMSQ*s

▪Last round, merge all compressed sets in the CS and all RS points into their nearest cluster

**Mahalanobis Distance**

Measure of the distance between a point P and a distribution D



**CURE Algorithm**

Pass 1

▪Pick a random sample of points that fit in main memory

▪Cluster these points hierarchically – group nearest points/clusters

▪Pick representative points  
-For each cluster, pick a sample of points, as dispersed as possible

-From the sample, pick representatives by moving them (say) 20% toward the centroid of the cluster

Pass 2  
▪Rescan the whole dataset and visit each point *p* in the data set

▪Place it in the “closest cluster”

▪Find the closest representative to *p* and assign it to representative’s cluster

**Girvan-Newman Algorithm**

▪Hierarchical divisive method  
▪Start with the whole graph  
▪Find edges whose removal “partitions” the graph ▪Repeat with each subgraph until single vertices

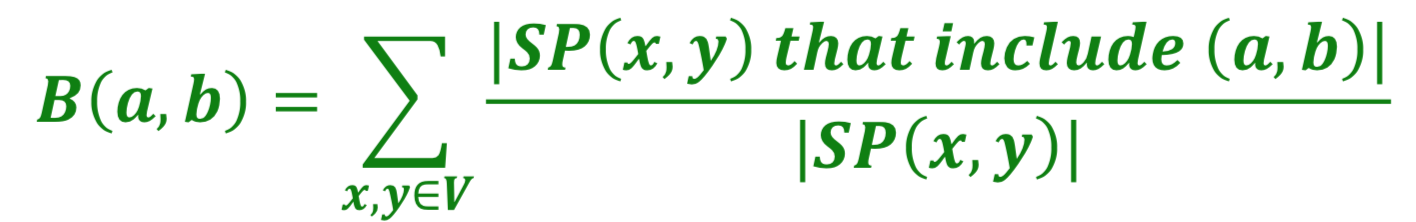
**Edge betweenness:** Number of shortest paths that pass through the edge

▪Betweenness of an edge (a,b) *B(a,b)*▪For each pair of nodes (x, y), compute number of shortest paths that include (a,b)  
▪May have multiple shortest paths between (x,y) *SP(x,y)*

Repeat the process for all nodes  
▪Edge betweenness is given by the sum of the flows

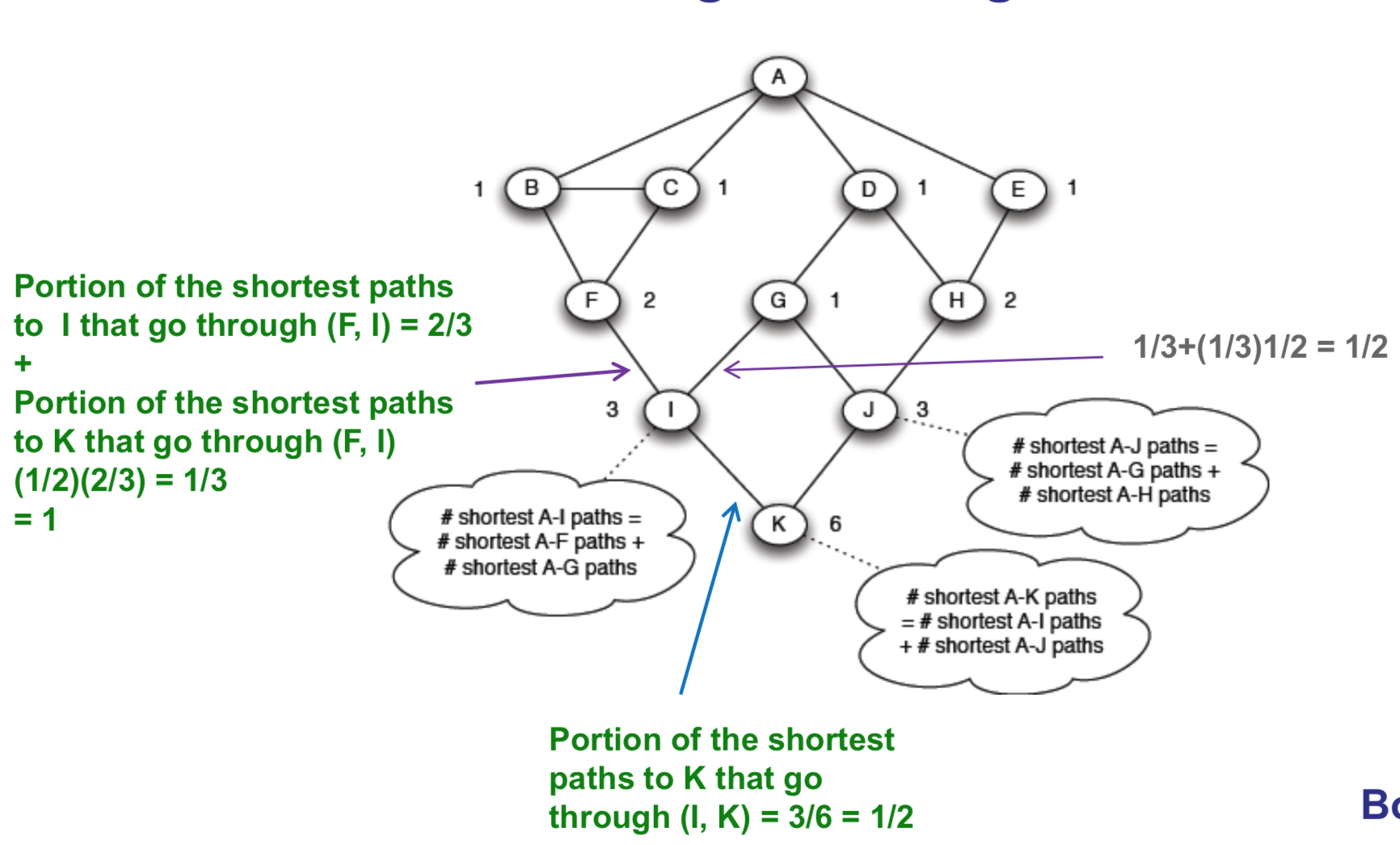
computed for each edge

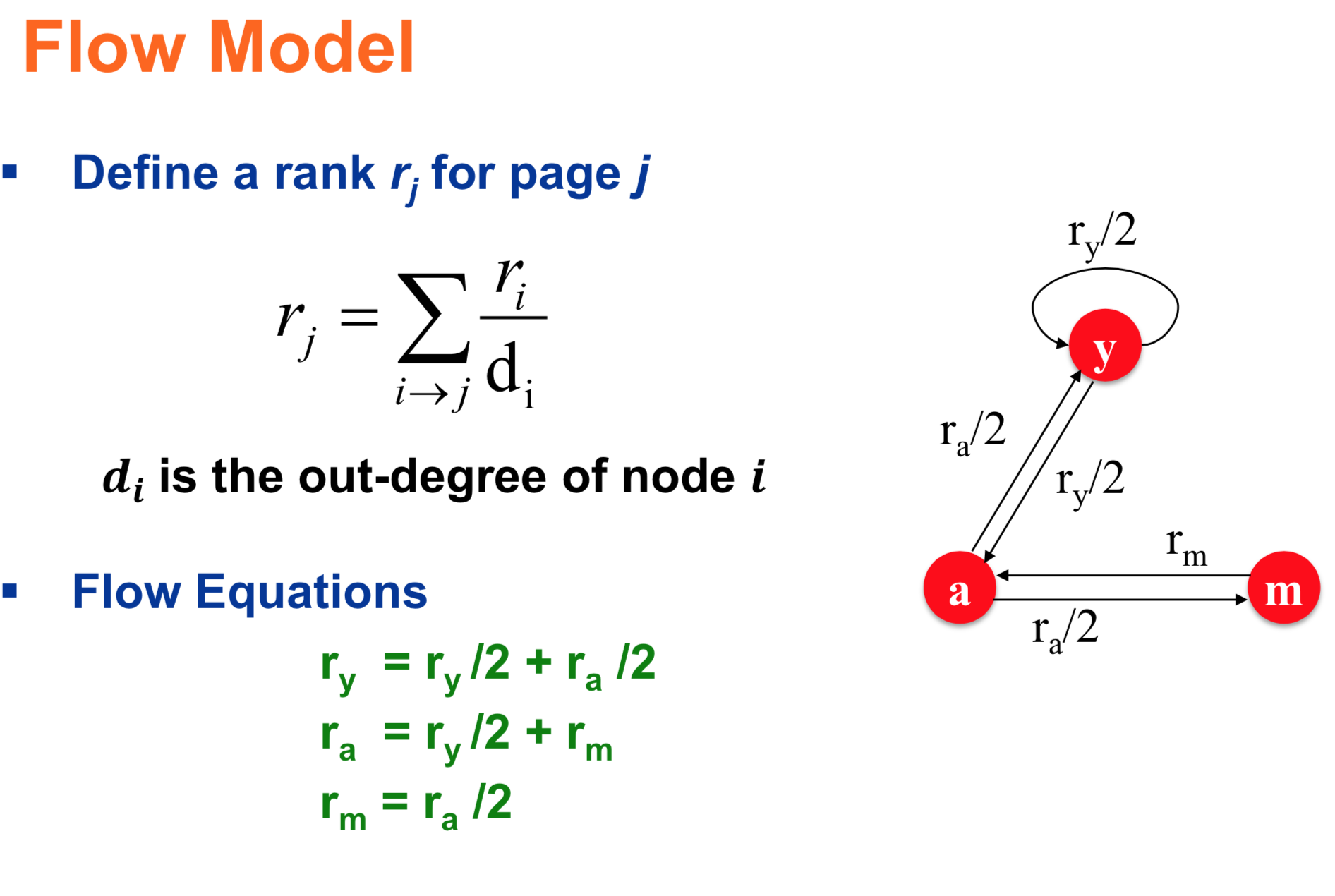
▪Compute the fraction of those that pass through (a,b)

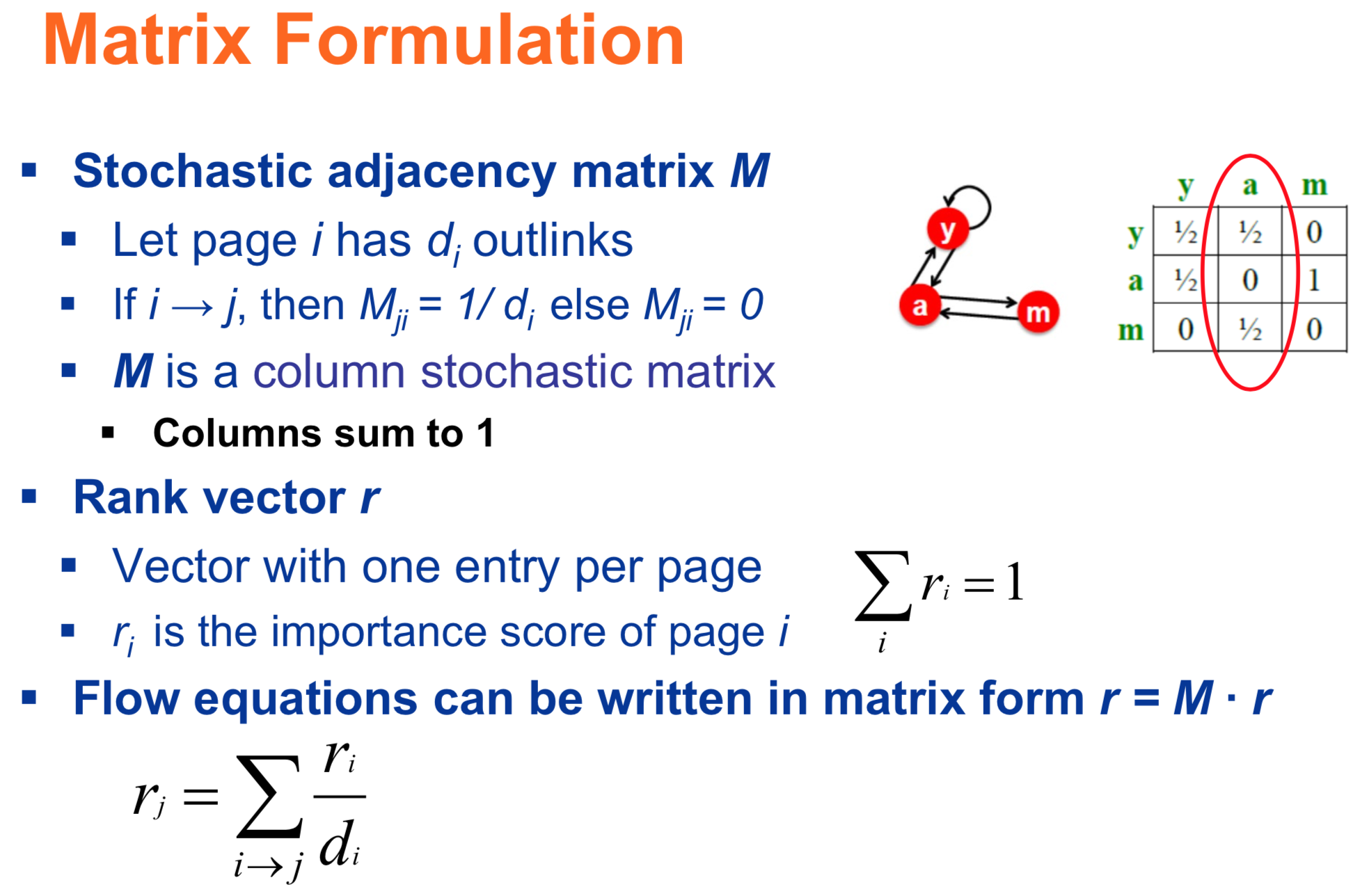


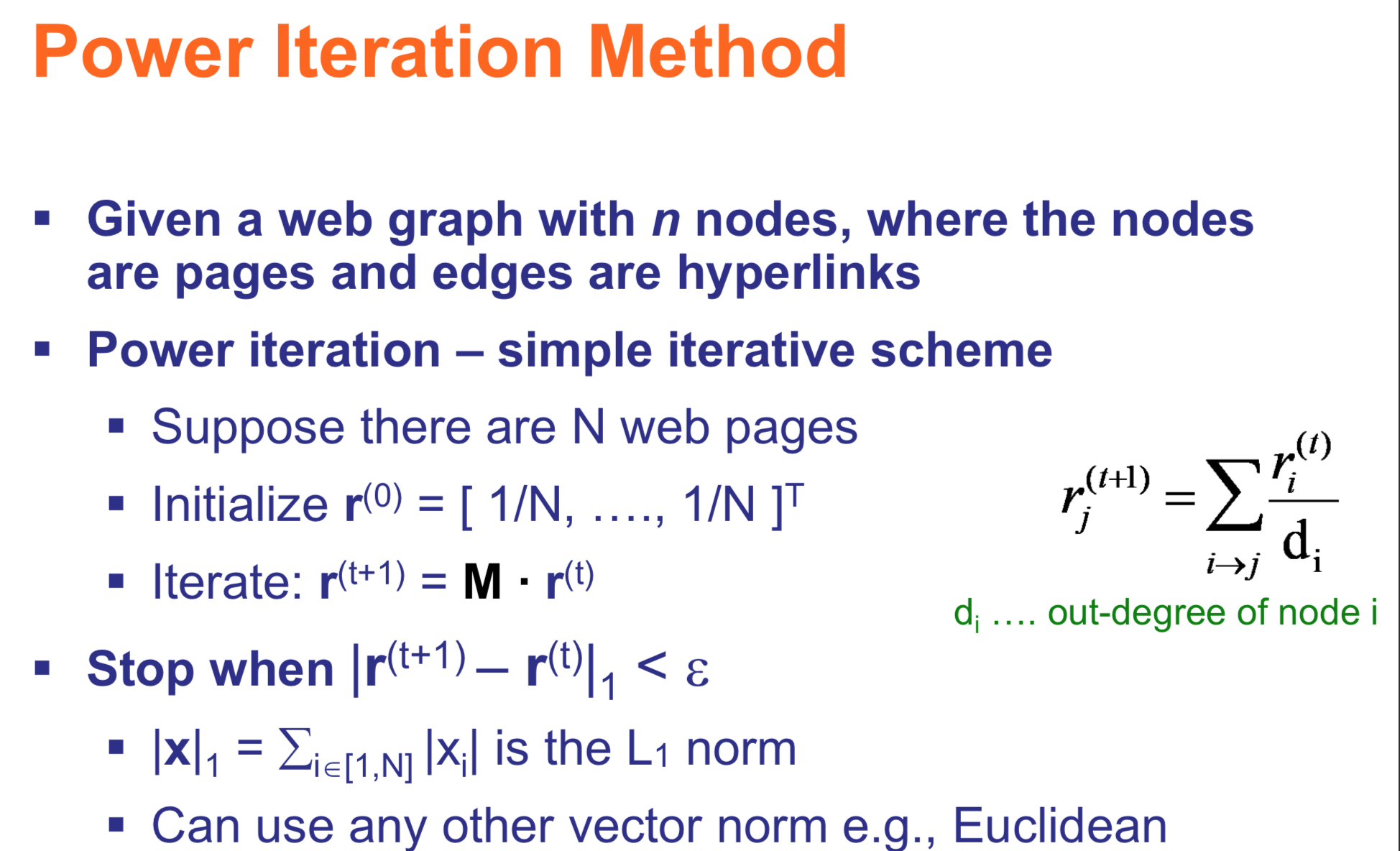
-Start with graph G.

▪Repeatedly remove edge with highest betweenness until <some stopping criterion>

▪Communities = resulting components 







Problem: Dead Ends

Solution: Teleport

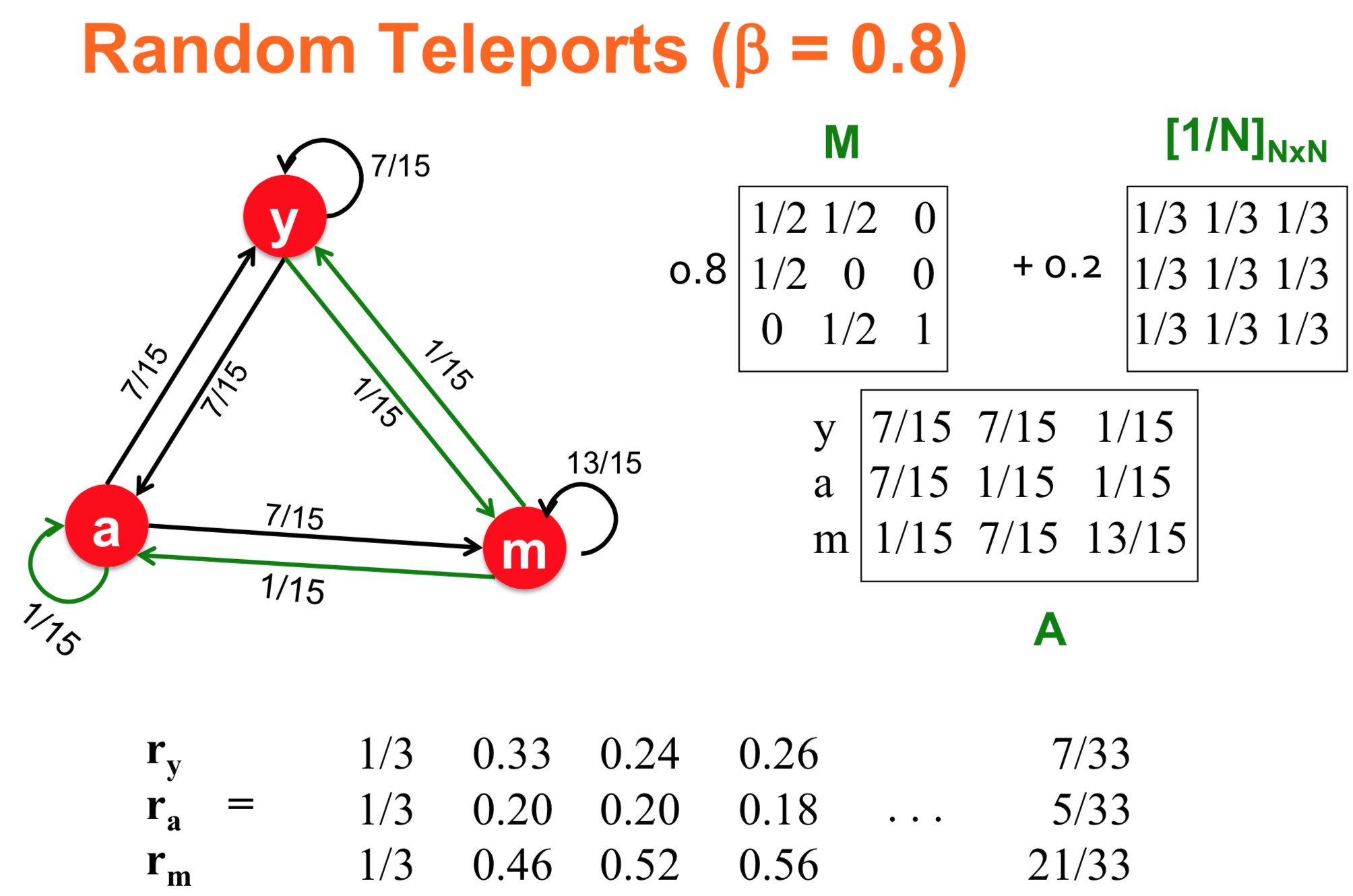
Adjust the matrix to allow a surfer to jump to some random page from dead ends

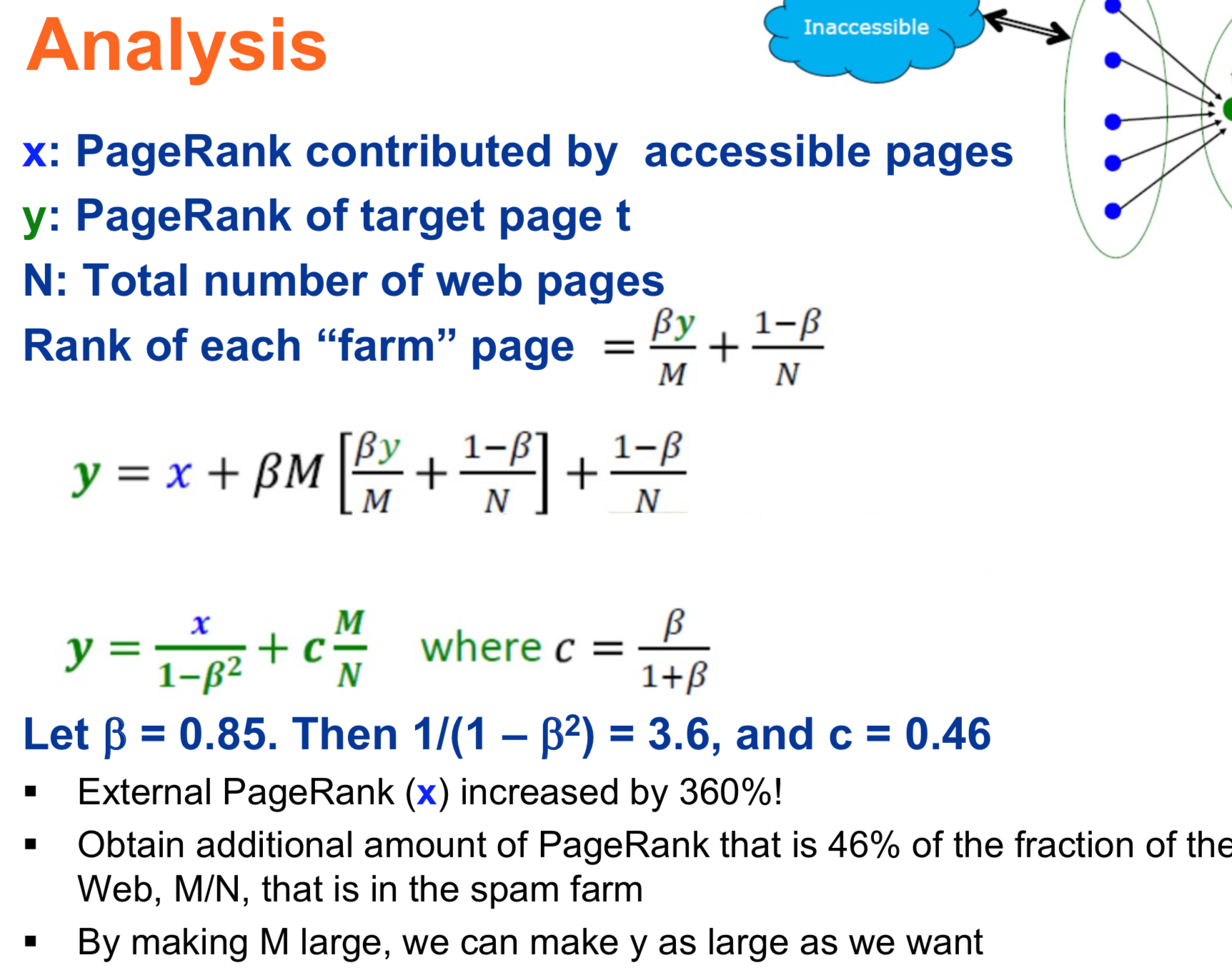
Problem: Spider Traps

Solution: Teleport

▪At each time step, a random surfer has two options ▪With probability β, follow a link at random  
▪With probability 1-β, jump to some random page  
▪Common values for β are in the range 0.8 to 0.9

▪Surfer will teleport out of spider trap within a few time steps





Trust Propagation (Simple Model)

▪Set trust of each trusted page to 1

▪Suppose trust of page *p* is *tp* ▪*p* has a set of out-links *op*

▪For each *q* ∈ *op*, *p* confers the trust to *q* ▪β *tp /|op|* for 0 <β < 1

▪Trust is additive  
▪Trust of *p* is the sum of the trust conferred on *p* by all its in-

linked pages

▪Trust attenuation  
▪Degree of trust conferred by a trusted page decreases with

the distance in the graph

▪Trust splitting  
▪The larger the number of out-links, the less scrutiny the

page author gives each out-link; trust is split across out-links

Hubs and Authorities

Each page has 2 roles, hence 2 scores (1) Authority

* ▪ Pages that contain useful information (quality in providing content) e.g., course home pages
* ▪ Voted or linked by “experts” (hubs)
* ▪ A good authority is linked from many good

hubs

(2) Hub

* + ▪ Pages that link to authorities (quality as an expert) e.g., list of course home pages
  + ▪ Votes of authorities pointed to
  + ▪ A good hub links to many authorities

Normalize it! HITS converges to a single stable point

1. What is the value of an in-link from *u* to *v*?
   * ▪ In the PageRank model, the value of the link depends on

the links into *u*

* + ▪ In the HITS model, it depends on the value of the other links out of *u* (vulnerable to spam)