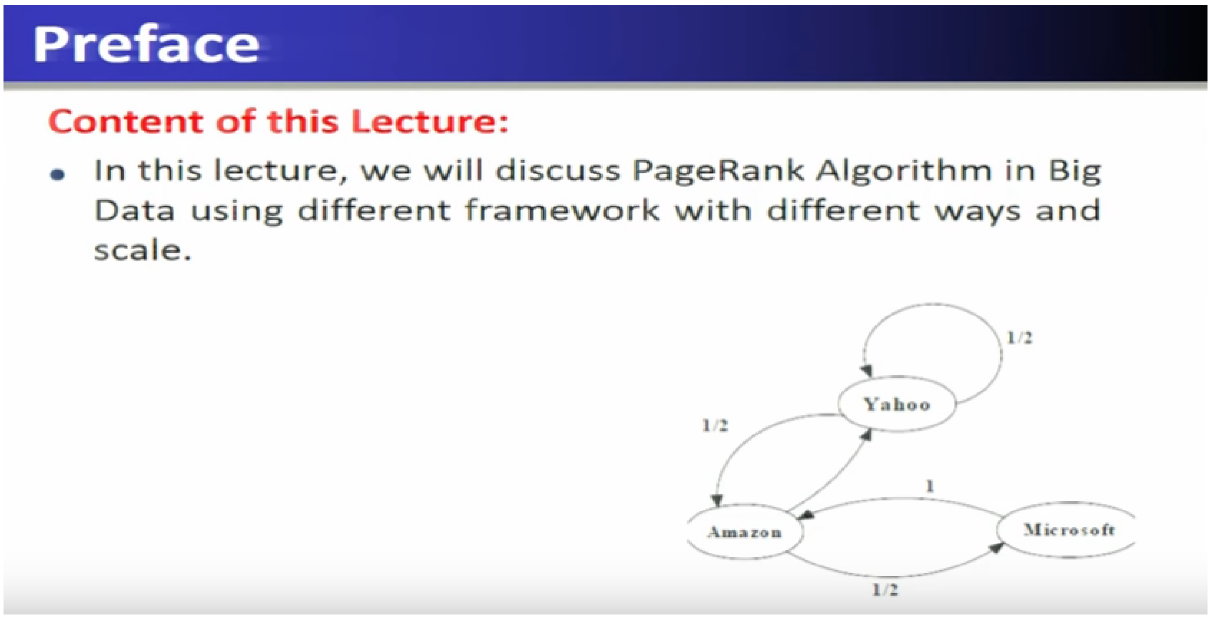
Lecture 31

PageRank Algorithm in Big Data

PageRank algorithm in Big Data.

Refer Slide Time :(0: 17)



Preface

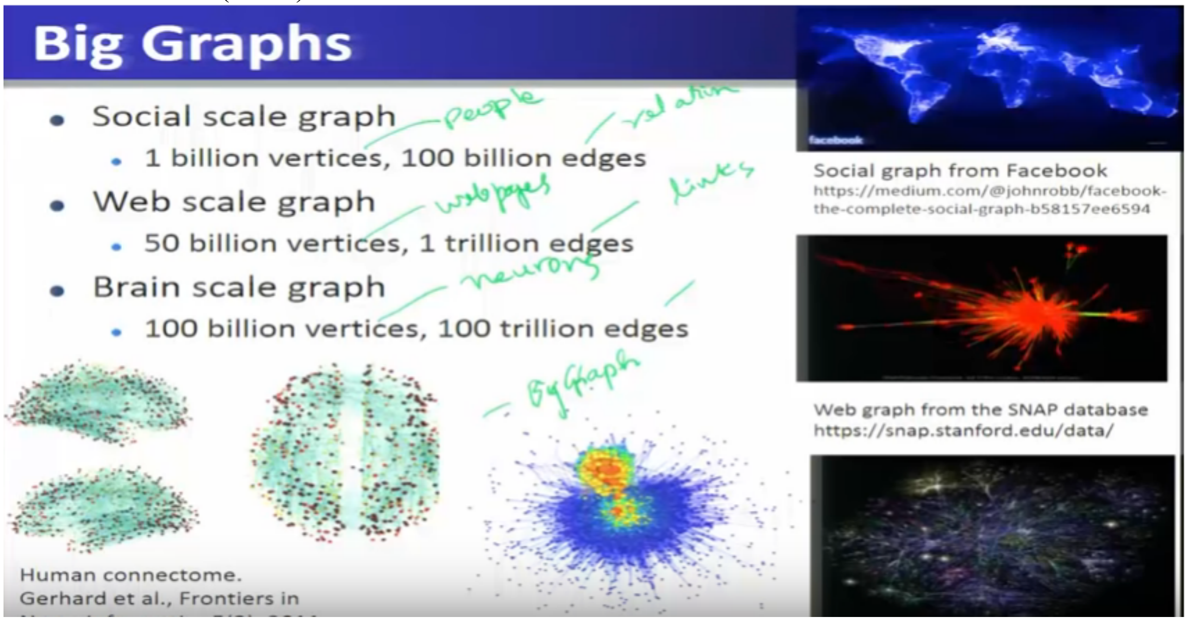
Content of this Lecture:

- In this lecture, we will discuss PageRank Algorithm in Big Data using different framework with different ways and scale.

1/2
Yahoo
1/2
1
Amazon
Microsoft
1/2

Preface content of this lecture, in this lecture we will discuss, PageRank algorithm, in big data using different framework, with different ways and the scale.

Refer Slide Time :(0: 32)



Big Graphs

- Social scale graph
  - 1 billion vertices, 100 billion edges
- Web scale graph
  - 50 billion vertices, 1 trillion edges
- Brain scale graph
  - 100 billion vertices, 100 trillion edges

Human connectome.
Gerhard et al., Frontiers in

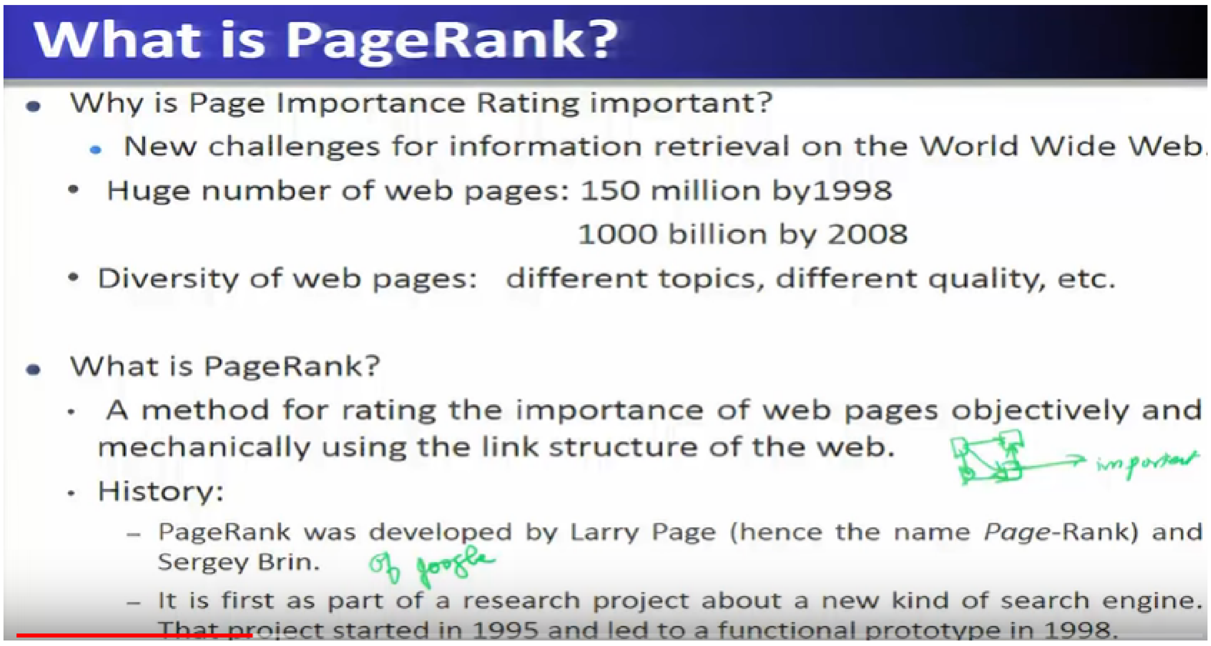
Social graph from Facebook
https://medium.com/@johnrobb/facebook-the-complete-social-graph-b58157ee6594

Web graph from the SNAP database
https://snap.stanford.edu/data/

Let us consider the big graphs, so in the social scale graph, you can see, 1 billion of vertices and 100 million of edges. So, vertices represents the set of people and the edges represents, the relations. So, in a socially scale graph, so billion of people, are interacting with each other, therefore the social scale graph consists of 1 billion vertices and 100 million edges. In a Webby scale graph which, contains the webpages, as the vertices, are much more than the number of people which are there, in the universe, hence 50 billion vertices that is the webpages, are there in web scale graph and one trillion edges.

Similarly in a brain scale graph that is, neurons, in a human brain, is much more than either the social scale graph or wavy scale graph: that is 100 billion of vertices, are there which are neurons and the 100 trillion edges, are the links between them, connecting those neurons. So, what is there in these graph is that, they are very, big size, therefore the computation of these very big huge graphs, is becoming a challenge and in this particular part of the discussion we will see, how we are going to process and do the computation, on such a big graphs.

Refer Slide Time :(2: 42)

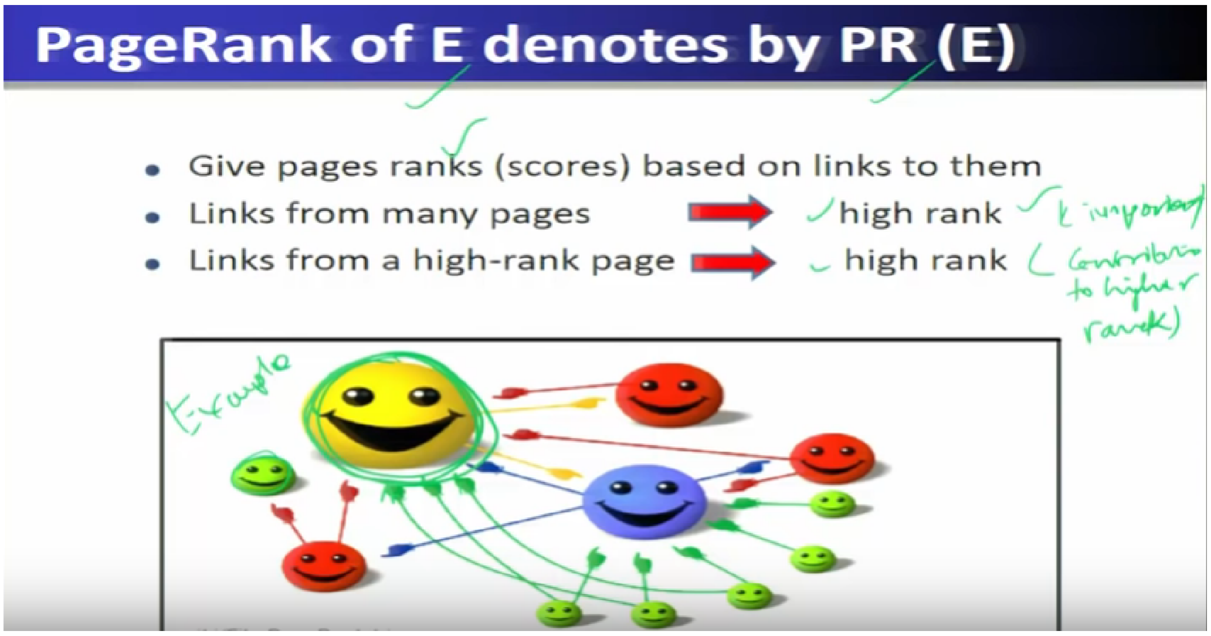


What is PageRank?

- Why is Page Importance Rating important?
  - New challenges for information retrieval on the World Wide Web.
  - Huge number of web pages: 150 million by1998
    1000 billion by 2008
  - Diversity of web pages: different topics, different quality, etc.
- What is PageRank?
  - A method for rating the importance of web pages objectively and mechanically using the link structure of the web.
  - History:
    - PageRank was developed by Larry Page (hence the name *Page*-Rank) and Sergey Brin.
    - It is first as part of a research project about a new kind of search engine. That project started in 1995 and led to a functional prototype in 1998.

One such algorithm which is quietly, which is well used is called a, 'PageRank Algorithm'. And PageRank algorithm on using Hadoop technology, is useful for computation on such a graph, of very big size. So, let us understand, about this particular algorithm, which is now, available, which is fairly used in, such a big graph in using this Hadoop technology. So, what we see here is that? The PageRank, which is also called as a, 'Ranking of the Pages' is why important, because, the new challenges of information retrieval, on the World Wide Web. Is there and it is very, common in use in most of the applications and the services. So, which considers a huge number of web pages: that is on the World Wide Web: that is 150 million, by 1998 and 1008. So, this particular graph, which is considered, for this computation, is going to be very, very huge and big. So, diversity of web pages, varies between different topics, different quality and so on. Now let us see, what do you mean by PageRank? Page Rank is a method for rating the importance of webpages, objectively and mechanically, using the link structure of the web, meaning to say that, the one might where if it is represented, in the form of the graph, of web pages, where the pages which are linked to each other, which are referring to each other, are having the link. So now, the page, which is being linked most of the other pages, is going to be very important page, whereas the other pages, which are not that much, linked by the other web pages are not that important. So, the method for rating the importance of web pages, using this link structure of the web, is called as the, 'PageRank'. So, the history of the PageRank, was can be can be traced back. So, the PageRank algorithm was developed by Larry Page and Sergey and Sergey Brin, of Google. It was it is the first as part of the research project, about the new kind of search engine: that where this Page rank algorithm was used: that project started 1995 and led to the functional prototype in 1998.

Refer Slide Time :(5: 40)



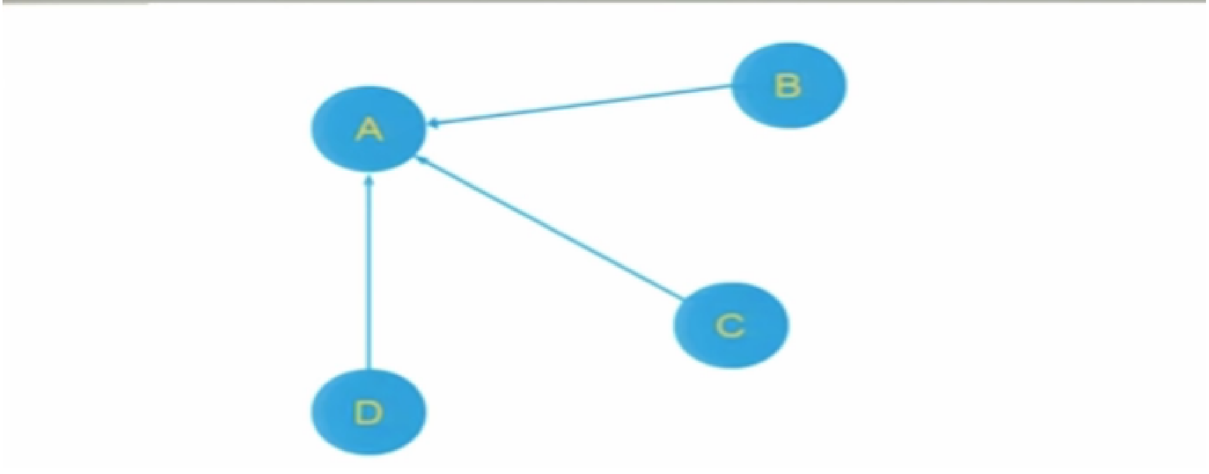
PageRank of E denotes by PR (E)

- Give pages ranks (scores) based on links to them
- Links from many pages $\rightarrow$ high rank (important
- Links from a high-rank page $\rightarrow$ high rank (Contribution to higher rank)

So, let us see that, using this particular picture. So, the pages which are very important, are shown with a, with a, with the bigger size, compared to the pages, which are not that important, shown by the smaller size. Now, the page rank is denoted by e, let us say, so the page rank we are denoting by the symbol PR. PR is the page rank of the page e. So give, this page rank algorithm will give, the pages the ranks, the score based on the, the number of links which are pointing to it. For example, this particular, this particular page which is shown, as a very fat, node you see that, most of these links are pointing to it. Therefore the size or the rank or the score of this particular page is higher, compared to the all other pages, we are not that many number of links are pointing to it. And if this particular, fat page, is pointing to some other web page, then that page also, get more importance, hence this kind of ranking, is computed through an algorithm, which gives the ranks or to the pages or this is also called as a, 'Score' which is fairly, based on the number of links, which are pointing to them. So, the number of links, from many pages, will indicate that it is a very high rank, web page or it is an important page and the links from a high, rank page is also, going to be contributing to a high rank. So, considering these two aspect or the rating, into the algorithm, the PageRank has combined these two different notions, to give, the rating or the score, to the webpages.

Refer Slide Time :(7: 57)

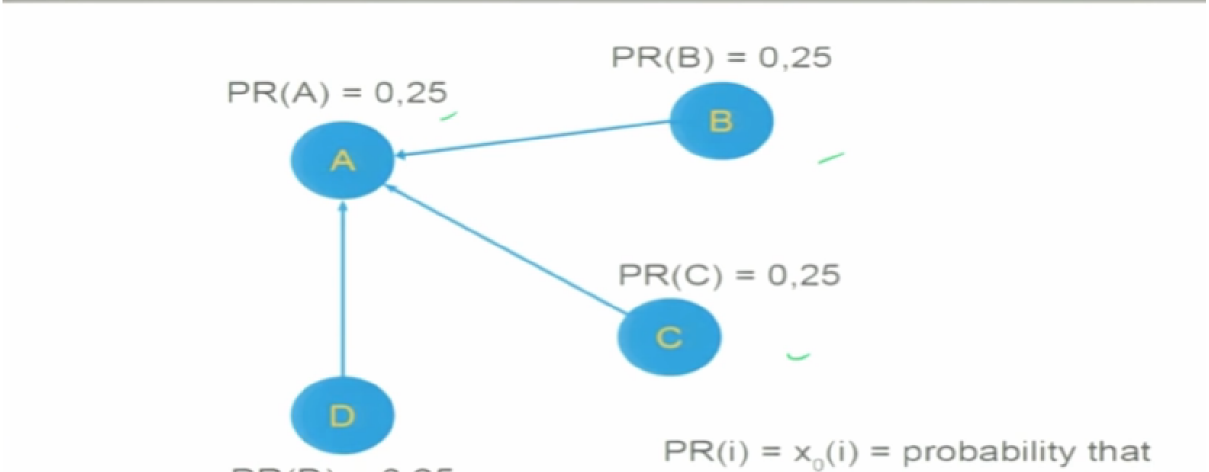
Example



Now, let us go in more detail of an example, let us say that, node A, is pointed by all other nodes that is BCD. Now, then in that case, the initial PageRank here, there are four nodes, so initial PageRank is equally divided 1 upon, 4 that is 0.25.

Refer Slide Time :(8: 22)

Example



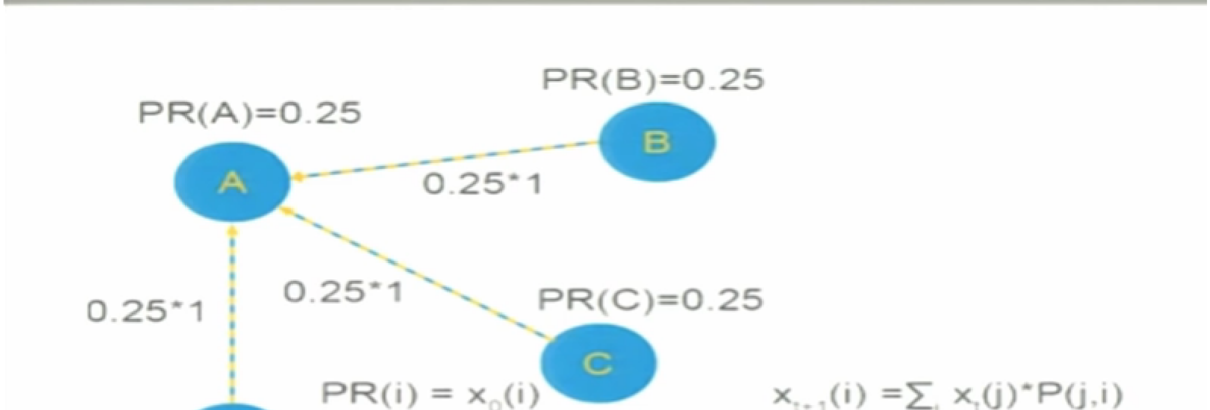
$PR(i) = x_0(i)$ = probability that the surfer is at node i at time 0

initial page rank $= \frac{1}{4}$ ✓

So, 0.25 is given the initial PageRank of every page: that is the initial page rank is equal to 1 upon, 4, for all the nodes, because the total number of nodes is 4 and let us say that, initially, it is to be divided 1 by 4. So, 0.25 is the initial page rank. Now, these values, are to be given back, to the page rank, so page rank of a will be recomputed again in the first iterations.

Refer Slide Time :(9: 02)

Example



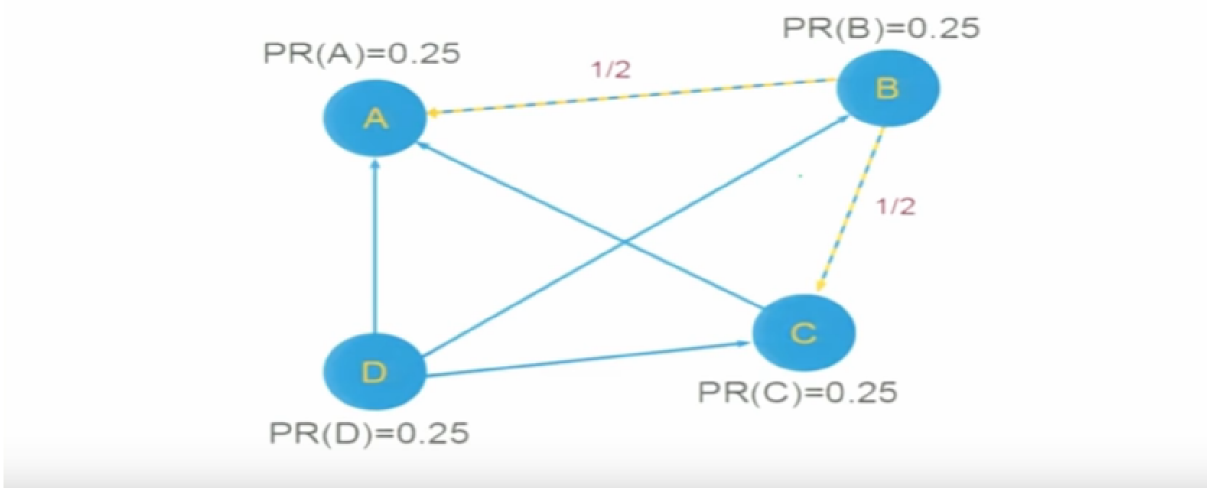
$PR(i) = x_0(i)$ $x_{t+1}(i) = \sum_j x_t(j) * P(j,i)$

new $PR(A) = x_1(A) = \sum_j x_0(j) * P(j,A) =$
$= x_0(B) * P(B,A) + x_0(C) * P(C,A) + x_0(D) * P(D,A) =$
$= PR(B) * 1 + PR(C) * 1 + PR(D) * 1 = 0.75$

So, after completing the first iteration, the, the page rank, of a will be recalculated in this manner.

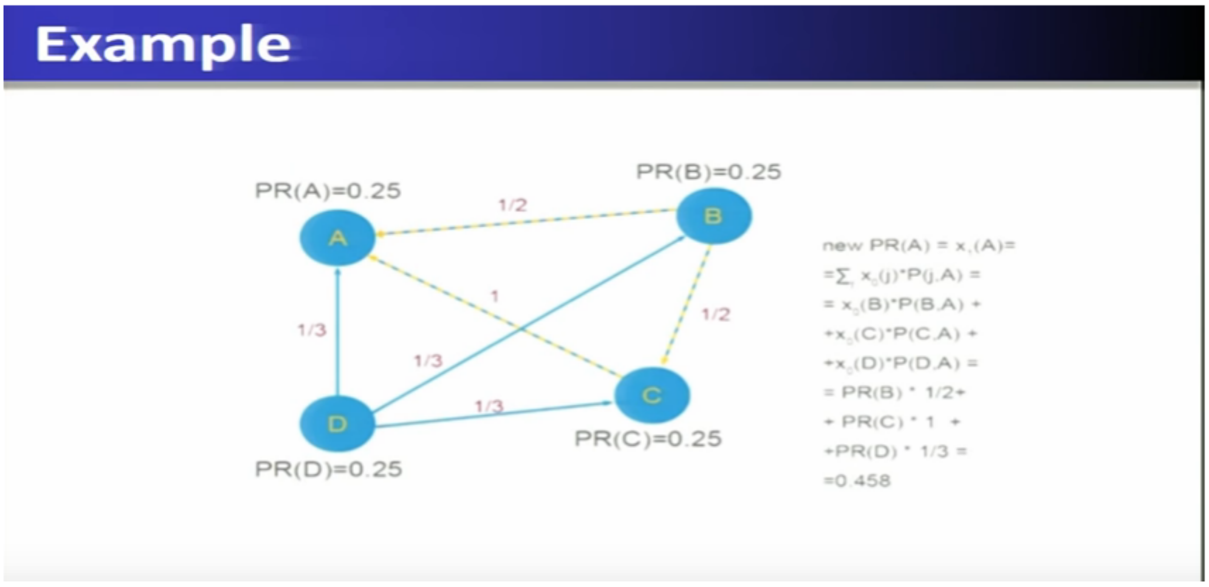
Refer Slide Time :(9: 13)

Example



So, for example, if let us say, B has also added the links, to a and C, so the, the page rank of B that is 0.25, will be equally divided into two parts and this page rank will be given back, to C and A.

Refer Slide Time :(9: 35)

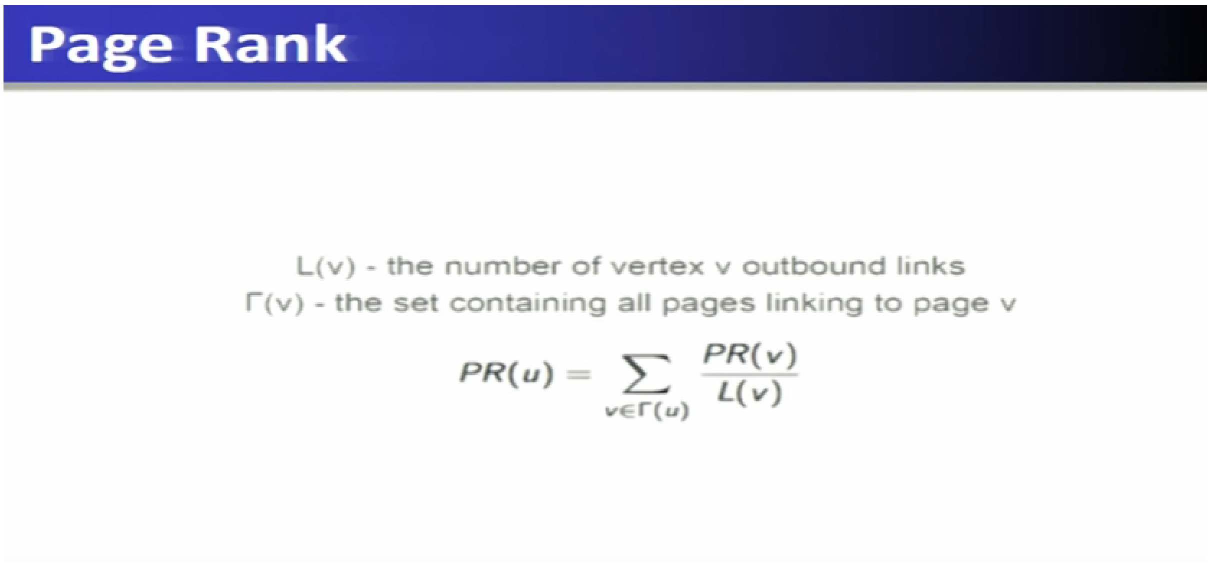


Example

new PR(A) = x_1(A)=
=$\sum_j x_0(j)$*P(j,A) =
= x_0(B)*P(B,A) +
+x_0(C)*P(C,A) +
+x_0(D)*P(D,A) =
= PR(B) * 1/2+
+ PR(C) * 1 +
+PR(D) * 1/3 =
=0.458

So, if we calculate it.

Refer Slide Time :(9: 40)



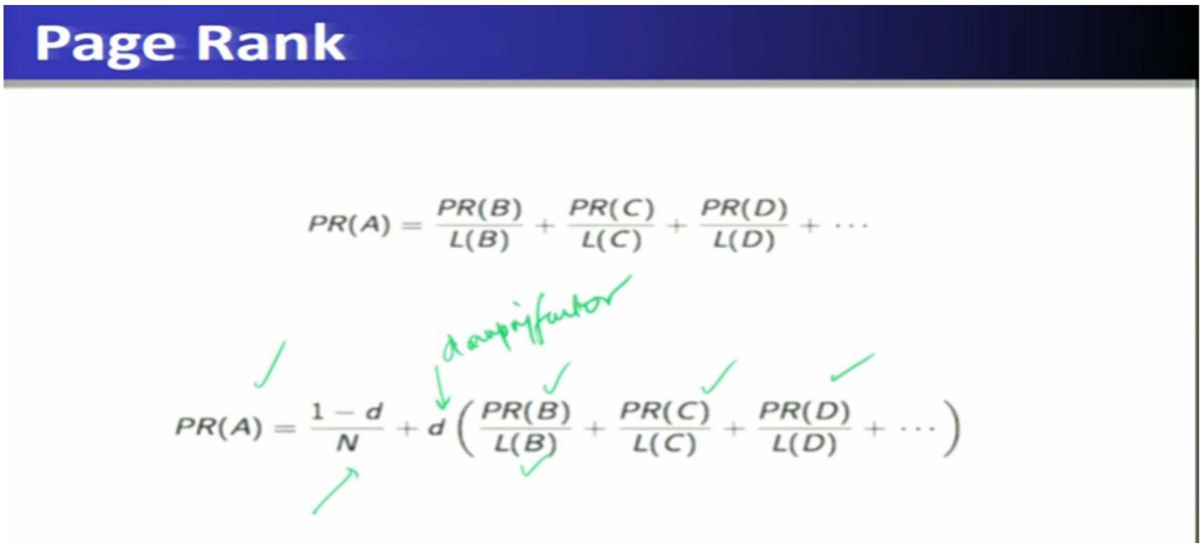
Page Rank

L(v) - the number of vertex v outbound links
Γ(v) - the set containing all pages linking to page v

$$PR(u) = \sum_{v \in \Gamma(u)} \frac{PR(v)}{L(v)}$$

In this manner and the page ranks, values, which are calculated, which is iterated in each iterations in this particular manner?

Refer Slide Time :(9: 51)



Page Rank

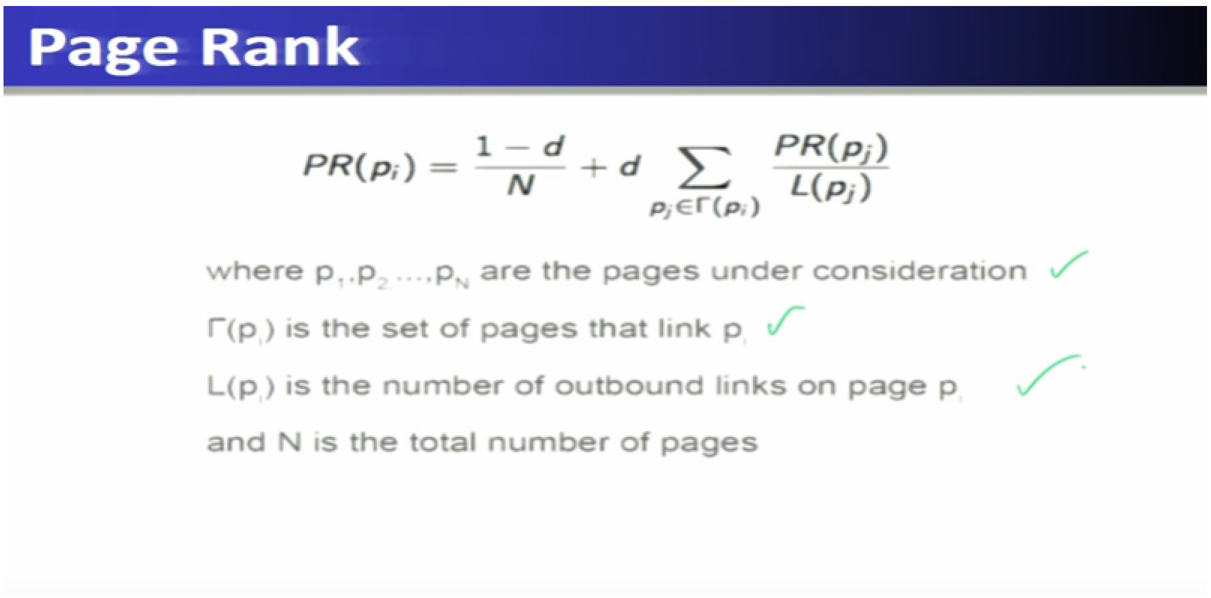
$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots$$

damping factor

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right)$$

So, PageRank value, is given by this particular formula, where each page for example, page A is pointed by page B page, C page, D and so on. So, their page rank divided by total number of outgoing links, which are there, it is to be divided, into equally and that is to be multiplied by the damping factor, Plus this 1 minus D upon n, n is the total number of pages.

Refer Slide Time :(10: 30)



Page Rank

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \frac{PR(p_j)}{L(p_j)}$$

where $p_1, p_2 \ldots, p_N$ are the pages under consideration ✓

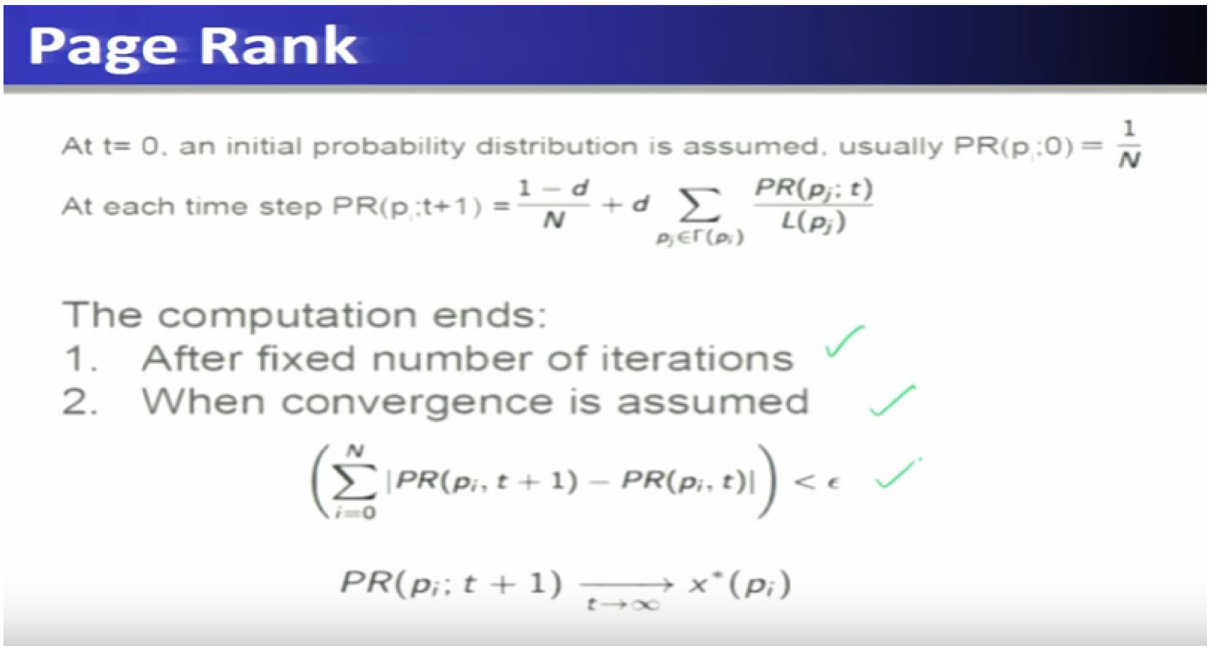
$\Gamma(p_i)$ is the set of pages that link p_i ✓

$L(p_i)$ is the number of outbound links on page p_i ✓

and N is the total number of pages

So, this particular way the page rank is calculated using this particular formula where P 1, P 2 and so on P and are the pages under the consideration and Rho P is the, set of pages that link, to the P and L P is the number of outbound links on page P and n is the total number of pages.

Refer Slide Time :(10: 55)



Page Rank

At t= 0, an initial probability distribution is assumed, usually $PR(p_i;0) = \frac{1}{N}$

At each time step $PR(p_i;t+1) = \frac{1-d}{N} + d \sum_{p_j \in \Gamma(p_i)} \frac{PR(p_j;t)}{L(p_j)}$

The computation ends:

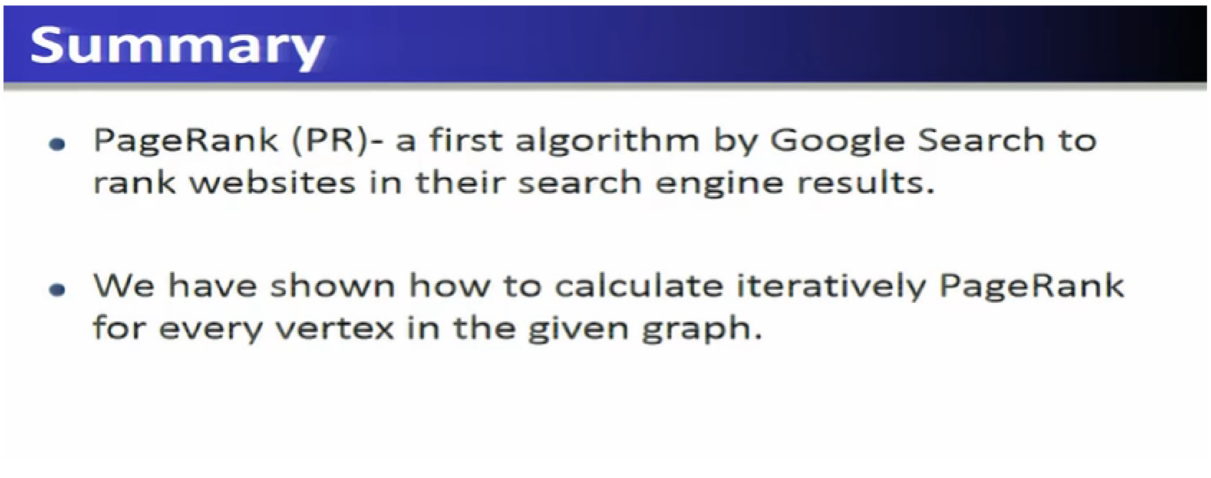
1. After fixed number of iterations ✓
2. When convergence is assumed ✓

$$\left(\sum_{i=0}^{N} |PR(p_i, t+1) - PR(p_i, t)|\right) < \epsilon \quad ✓$$

$$PR(p_i; t+1) \xrightarrow[t \to \infty]{} x^*(p_i)$$

Now, this particular, after the fixed number of iterations or when the convergence, is assumed, to be a very small value, of changes then the,

Refer Slide Time :(11: 08)

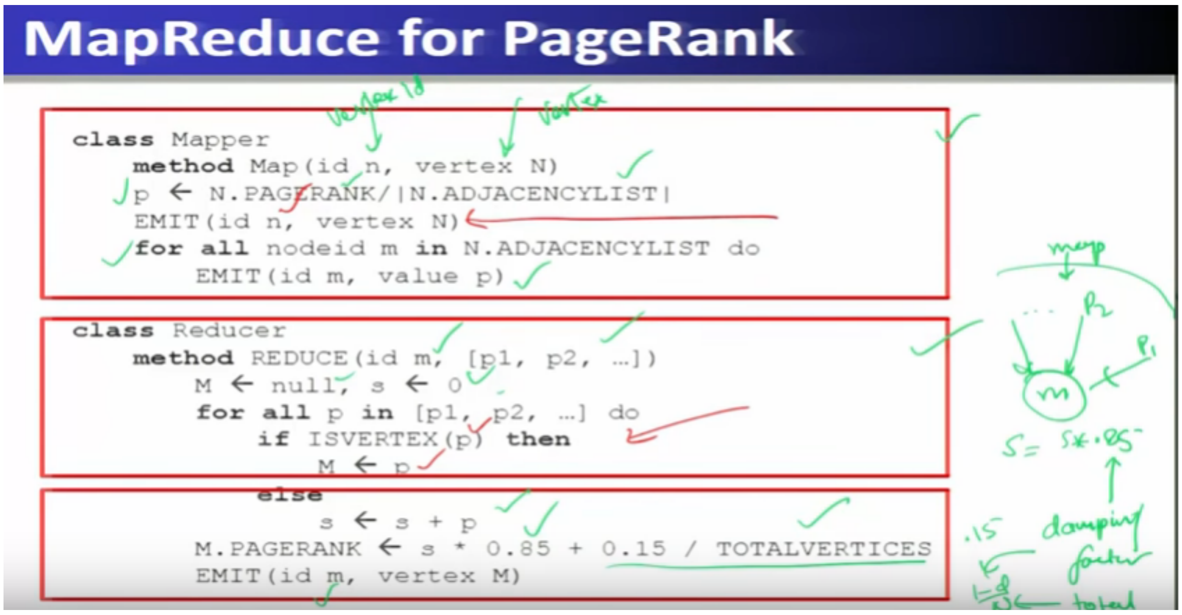


Summary

- PageRank (PR)- a first algorithm by Google Search to rank websites in their search engine results.
- We have shown how to calculate iteratively PageRank for every vertex in the given graph.

then the execution of the PageRank algorithm, converges and the final values of the page rank, we the page rank of those algorithm. So, in summary the page rank is the first algorithm, given by the Google search engine, to rank the web pages, in their search engine results. So, we have shown how to calculate iteratively the page rank using formula for every, vertex in a given graph.

Refer Slide Time :(11: 35)



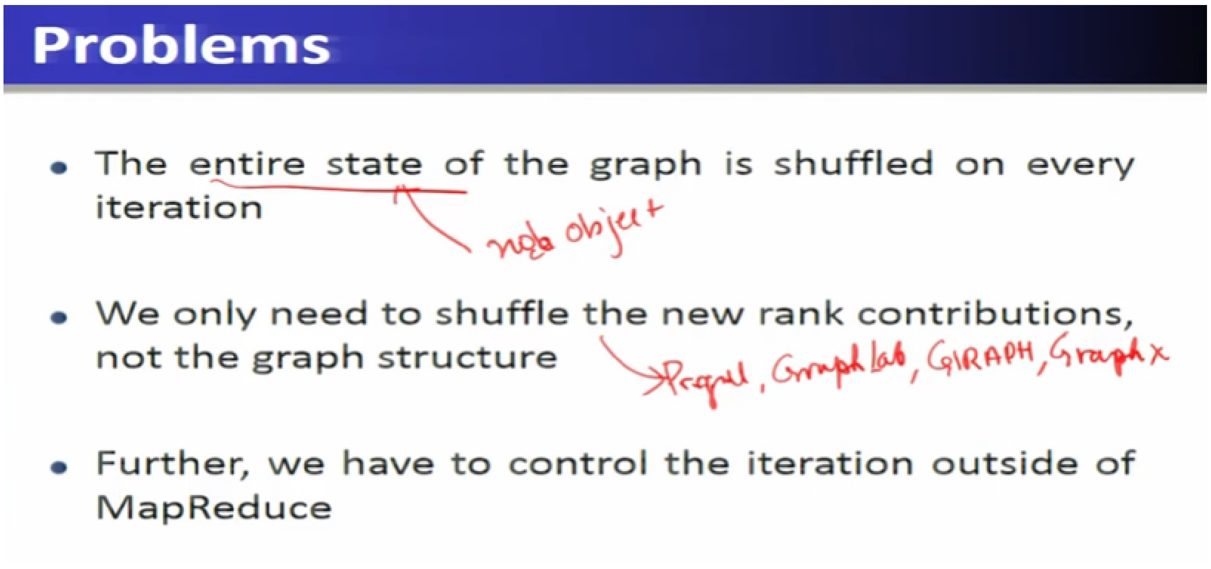
MapReduce for PageRank

```
class Mapper
    method Map(id n, vertex N)
    p ← N.PAGERANK/|N.ADJACENCYLIST|
    EMIT(id n, vertex N)
    for all nodeid m in N.ADJACENCYLIST do
        EMIT(id m, value p)

class Reducer
    method REDUCE(id m, [p1, p2, …])
        M ← null, s ← 0
        for all p in [p1, p2, …] do
            if ISVERTEX(p) then
                M ← p
            else
                s ← s + p
        M.PAGERANK ← s * 0.85 + 0.15 / TOTALVERTICES
        EMIT(id m, vertex M)
```

Let us see how, Map Reduce can be applied to this PageRank algorithm, so that large scale graphs, we can apply this algorithm PageRank on a very big size crops. So, PageRank may produce applied on a page rank has, two components one is called, 'Mapper' the other is called the, 'Reducer'. In the mapper phase, we have taken the input as the node IDs and so, it is the vertex ID and this is the vertex, for which we are going to calculate the PageRank. So, for a vertex and what we will do here is that? We will find out or we will consider the current page rank of node n and also, you will find out that, total number of adjacent, at the sensi list or the total number of neighbours, of n. So, this particular page rank will be divided by the total number of neighbours: that will be calculating its page rank. And then, for all values of the node IDs, M in its neighbour list or it's as NC list, it will emit, this values, of this ID, m and the value of P. Now, as far as, the reducer is concerned after getting these values M, for the node M, all these values, when it is collected: that is Page ranks from all its neighbours. So, like let us say that this, when it collects the page rank from different P 1, P 2 and so, on up to. So, these page ranks ment, is being sent by the map function and it when it is received at M, then the reducer will now, calculate the page ranks. So, it will start with M is equal to null and s, the calculating the value of page rank, in s variable. So, what it will do is, for all P values, which are there receiving the page ranks and if it is, not a vertex: that means a p is not a vertex, then it is the page rank which is being sent. So basically, it will add, the value of s to P and the page rank will be calculated, based on, the PageRank is equal to s, divided by 0.85. So, 0.85, s multiplied by 0.85, 0.85 is a damping factor and the other portion, where you have to add 1 minus D upon n, so n is the total number of vertices, total vertices and 1 minus D that is, 1 minus 0.85 it becomes, 0.15. So, this will be the total summation of these two will become the PageRank, of no DOM and it will be emitted, from the reducer function, in vertex ID. Now, there is one more function, one more statement which we have omitted is that, this not only will send the PageRank of the neighbouring node, but, the node itself, as the complex object will be emitted out here in this algorithm. So, it will emit N and vertex n, so here it will be checked, if P is the vertex, is P is the vertex, so if P is the vertex, then M will be, added to the P: that means it is the complete complex node object, which is being emitted out of the mapper, not the page ranks. So, both of them, are being used so P is the, the complex object is the state variable, which are also, transferred or emitted, along with the PageRank values, so this was the Map Reduce for implementation for PageRank.

Refer Slide Time :(16: 13)



Problems

- The entire state of the graph is shuffled on every iteration

node object

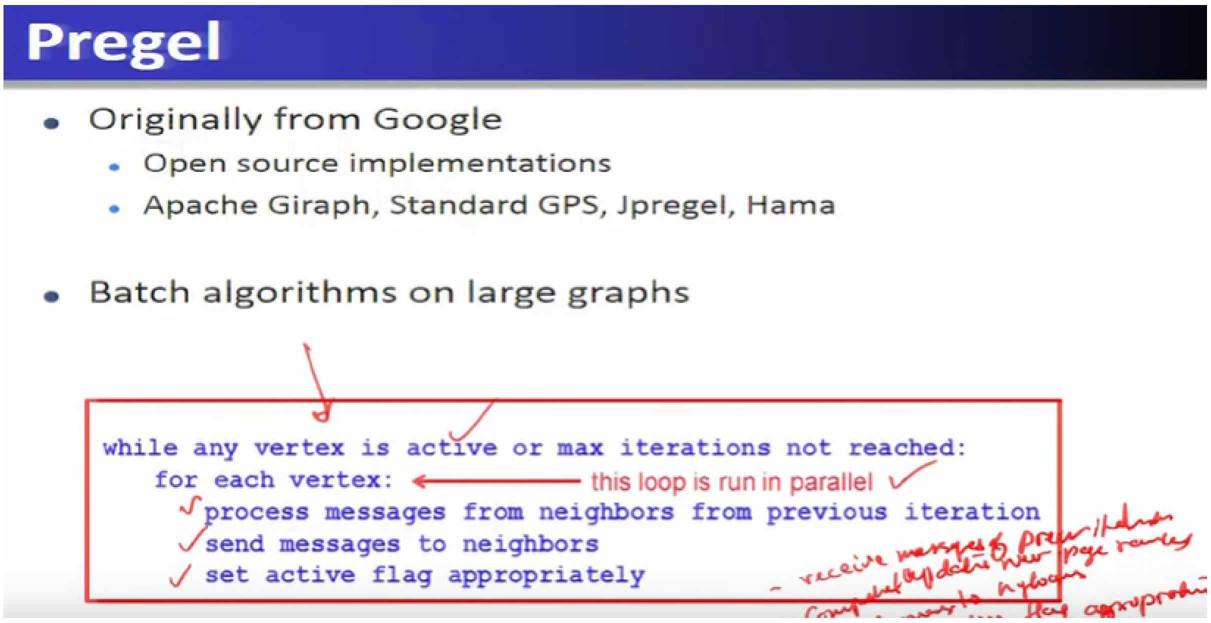
- We only need to shuffle the new rank contributions, not the graph structure

Pregel, GraphLab, GIRAPH, Graphx

- Further, we have to control the iteration outside of MapReduce

So, here let us see the problems, in this implementation of a Map Reduce version of a PageRank here we have see that, entire state of a graph is shuffled on every iteration. State in the sense we have seen some, the node object, as the state variable also, in along with the PageRank values, also is being communicated. So, we only need to suffer, the new rank contribution not the entire graph structure, so this is the problem of internal implementation, so in a newer versions, like Pringle, then Graph Lab or Giraph or graphics has given concept of the internal implementation, of the iterations of a graph, hence the transferring of the entire state, during the shuffle is not required, therefore more efficient versions, are now, available the initial of this implementation problem, which we have just seen. Further we only need to shuffle, the new rank contribution not the entire graph structure: that we have seen that in the newer, framework of a graph computation, this is resolved. Further we have to control the iterations outside of the Map Reduce.

Refer Slide Time :(17: 47)



Pregel

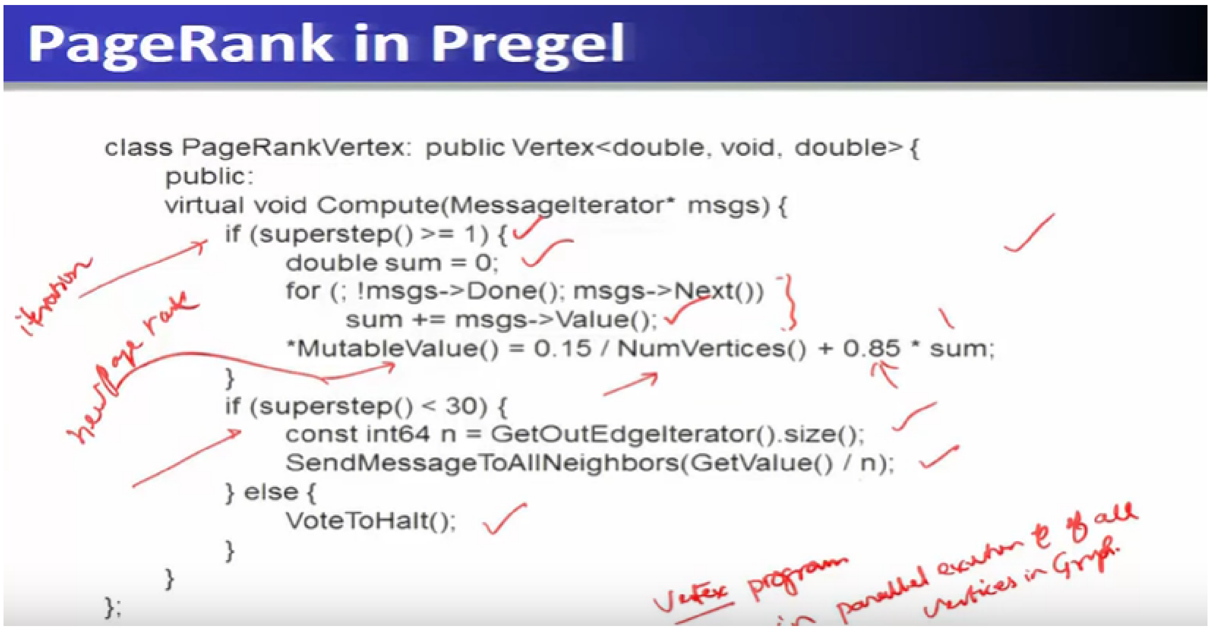
- Originally from Google
  - Open source implementations
  - Apache Giraph, Standard GPS, Jpregel, Hama
- Batch algorithms on large graphs

```
while any vertex is active or max iterations not reached:
    for each vertex:  ←—— this loop is run in parallel ✓
      ✓process messages from neighbors from previous iteration
      ✓send messages to neighbors
      ✓ set active flag appropriately
```

- receive messages of previous iteration
Compute/update new page rank
send message to neighbours
flag appropriately

Let us see how the pregel, the PageRank is implemented. Pregel, is originally from the Google it is an open source implementation, the other such open source implementation, which are available called, ‘Apache Giraph’ and standard GPS, then J Pregel and Hama. Batch algorithms on the large-scale graph, processing it is how is being used, the Pregel. Now let us see the, iterations in the Pregel, so Pregel all doesn't, use the Map Reduce notion, of the concept which we have seen in the previous slide. So, here in the Pregel, all the entire vertex, is being programmed, only one vertex program is written. And here, we have to see for any vertex, which is active or if the maximum iterations not reached, than for each vertex, this particular loop will, run in parallel, what it will do, in this iteration is that, it will process the messages, from the neighbours, from the previous iterations. So, this is required why because, it will bring the page ranks, from the neighbours and then, it will send the message, to the neighbours and set the active flag, appropriately. So, what it will do is, it will collect, it will, it will receive the messages, the messages of the previous iteration and compute or oblique update, the new page ranks and then again, send the message, to the neighbours, about the new PageRank and it will set, the active flag appropriately. So, that means when a particular node, is not doing any of these actions, then the node, is not active, its passive and it will be out of the action, in the further iterations.

Refer Slide Time :(20: 18)

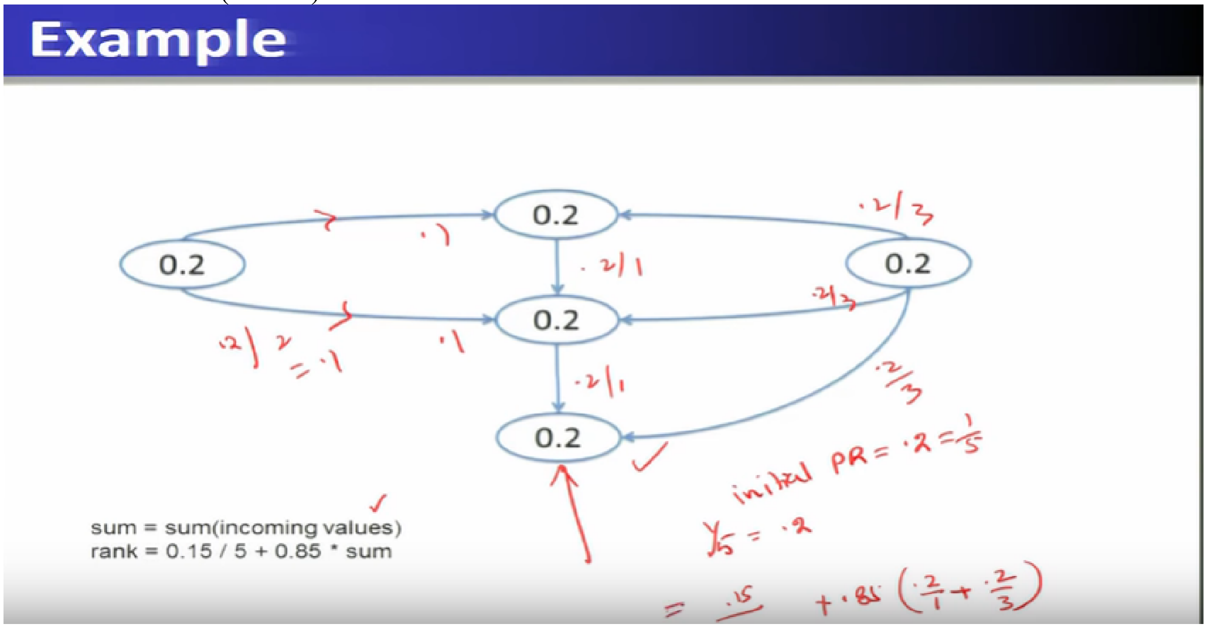


PageRank in Pregel

```
class PageRankVertex: public Vertex<double, void, double>{
      public:
      virtual void Compute(MessageIterator* msgs){
            if (superstep() >= 1) {
                  double sum = 0;
                  for (; !msgs->Done(); msgs->Next())
                        sum += msgs->Value();
                  *MutableValue() = 0.15 / NumVertices() + 0.85 * sum;
            }
            if (superstep() < 30) {
                  const int64 n = GetOutEdgeIterator().size();
                  SendMessageToAllNeighbors(GetValue() / n);
            } else {
                  VoteToHalt();
            }
      }
};
```

So, let us go and see the, detail algorithm of page rank in the pregel. So, here we, we have, we see that, it is a super step, super step means, this is the iteration, of the PageRank algorithm, on a particular vertex or a node, what it does it will start, it will initialize, the value of sum and it will collect, the messages, the values which comes into the messages and then using this particular summation of all the values, of the PageRank which it has collected, it will apply the damping factor that is 0.85 and then it will calculate 0.15 divided by total number of nodes and this will be the new PageRank. Now, if this particular, in number of iterations, is less than 30 and then, it will that means, it is not terminated, still it has to send the messages. So, it will send the messages, to all the neighbours and send to all the neighbours, divided by n why because, the, the PageRank will be equally divided, across all the outgoing edges. And this particular iterations will keep on repeating and in the end, it will go to the halt, if no action, is being taken. So, this is the vertex program, of the PageRank in prison, you see that, it is not becoming complicated as you have seen in the Map Reduce. So, a vertex program, we can write down and which will calculate the PageRank, in parallel, of all the vertices, in the graph. So, the parallel execution, parallel graph execution, for this particular code, will run on all the vertices and they will calculate the page rank or iterations and once the iterations, when's the iterations finishes: that is it converges, the PageRank and it will be the value.

Refer Slide Time :(22: 57)

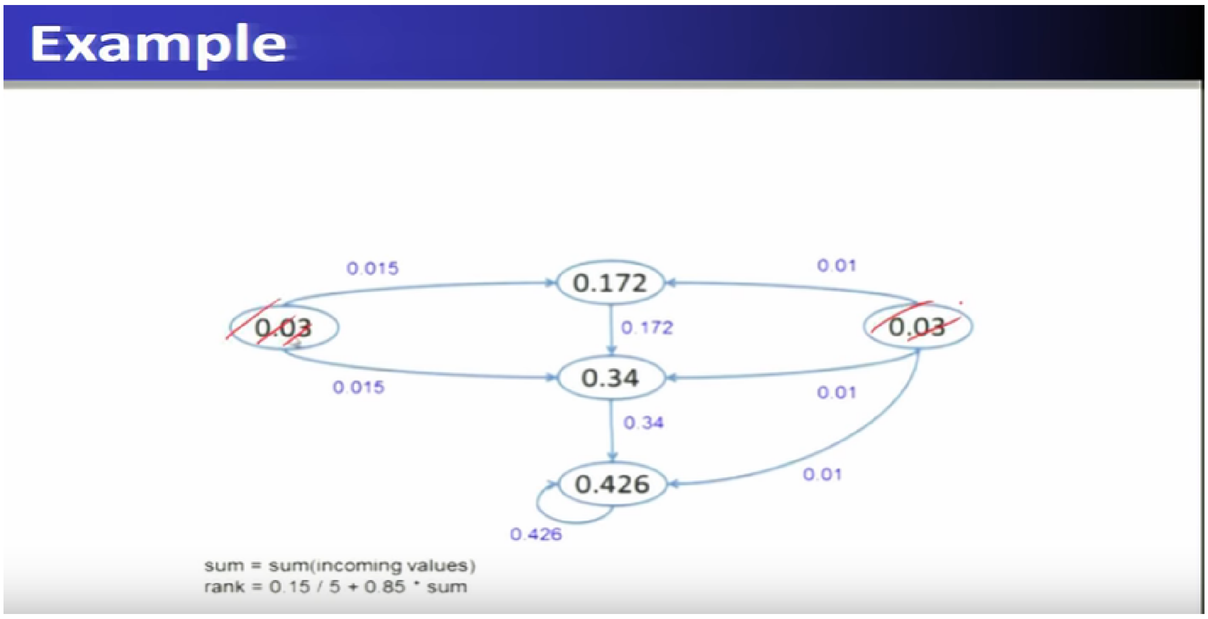


Example

sum = sum(incoming values)
rank = 0.15 / 5 + 0.85 * sum

So, this is the PageRank implementation. Let us see the example, of this algorithm and this approach. So, this is the initial condition, so initial page ranks, of all the roads will be, so there are five nodes, so 1 upon 5 that is 0.2. So, 0.2 will be the initial PageRank and that is 1 upon 5, it is given to all the nodes. Now let us see that, this particular node has 2 outgoing, 2 outgoing links, so hence, so it is 2 outgoing links, so basically the share, of the PageRank which will go along this link is, 0.2 divided by 2 that becomes 0.1. So, point 1 and point 1 will be the contribution, of the PageRank, on all these links similarly here, 1, 2, 3. So, point two divided by three, point two divided by 3, point 2 divided by 3, will be the PageRank which will be sent along that link. And here point two, will be there is only one outgoing edge, so the point two divided by 1 and here also, point two divided by 1, this is there. Now, as far as, this node is concerned, let us consider this node. So, what it does is, it will calculate, the sum of incoming page ranks: that is 0.2, divided by 1, plus 0.2 divided by 3. So and then it will multiply by point five that is, a damping ratio and plus, what it does, is that, it will add point one five, divided by total number of nodes, total number of nodes is: that is three, here in this case. So, in this manner the, the incoming,

Refer Slide Time :(25: 06)

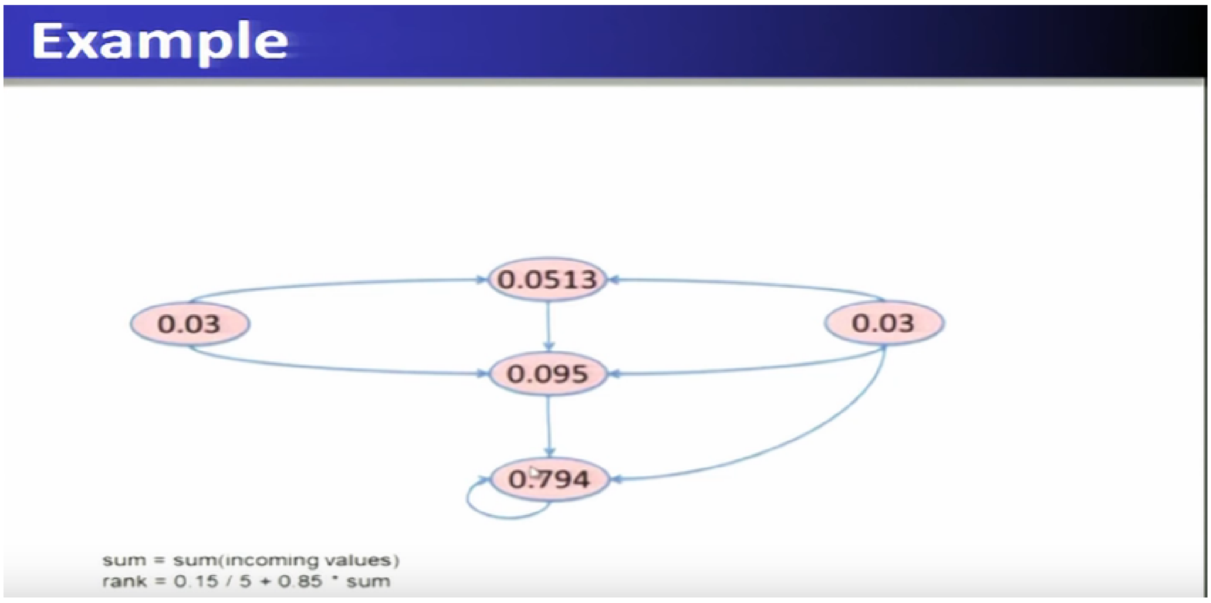


Example

sum = sum(incoming values)
rank = 0.15 / 5 + 0.85 * sum

so basically you can see here. The value will become, 0.42 six, over here and this is 0.1 and why these values are changed is because, this contribution is 0. So, 0.15 divided by five will become point zero three and this is the iteration number one and in the next iteration, you see that, these values, these two values are not changing in the next iteration.

Refer Slide Time :(25: 43)

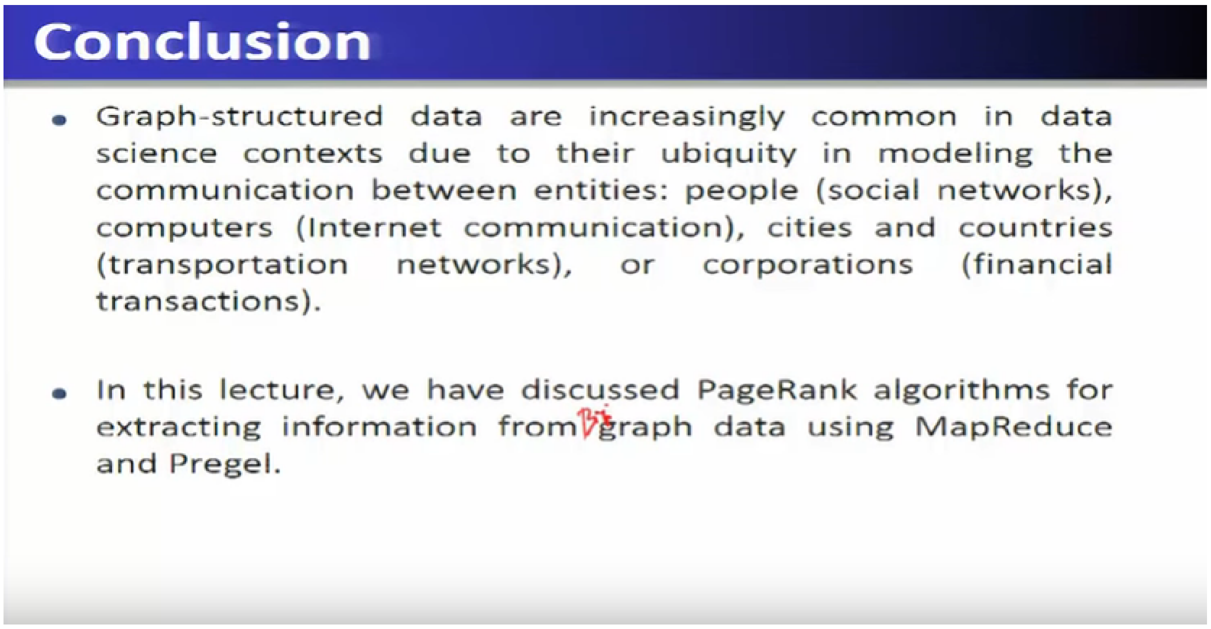


Example

sum = sum(incoming values)
rank = 0.15 / 5 + 0.85 * sum

So, these two values are not changing in the rate, in the next iteration, so they will become red. Red in the sense, they will stop participating in the further iterations and these values, have to be which are sent by these nodes earlier, they will be stored in the buffer, because in the in the further iteration they will not participate but, these values are, going to be fixed and a constant which is going to be used. And you see that, here now, in the in this iteration this particular node is, out of the action, so it will be not active and finally, at this stage the, the page rank algorithm, will terminate, with these values.

Refer Slide Time :(26: 30)



Conclusion

- Graph-structured data are increasingly common in data science contexts due to their ubiquity in modeling the communication between entities: people (social networks), computers (Internet communication), cities and countries (transportation networks), or corporations (financial transactions).
- In this lecture, we have discussed PageRank algorithms for extracting information from Big graph data using MapReduce and Pregel.

So, in conclusion graph structure data, are increasingly common in data, in data science context, due to their ubiquity in modeling the communication between entities: that is the people, in the social network computers, in the internet communication cities and countries in the transportation network. And the web page is no, word white web or the corporations, in the financial car operations. So, in this lecture, we have discussed important algorithm: that is a PageRank algorithm for extracting the information, of the big graphs, big graph data. Using the Map Reduce and using the pregel paradigm. Thank you.