**Instructions**

Q1. Build inverted index.

Input Split 1 (Doc ID 101) [ cat pat mat sat cat eat ]

Input Split 2 (Doc ID 201)  [ pat mat sat pat mat eat]

Input Split 3 (Doc ID 301) [ sat mat cat pat fat mat]

Assume three Mappers and two Reducers.

Let cat mat and sat be processed by Reducer 1 and rest by Reducer 2.

**Mappers Output:-**

**Input Split 1 Input Split 2 Input Split 3**

((cat ,101),2) ((pat ,201),2) ((sat ,301),1)

((pat ,101),1) ((mat ,201),2) ((mat ,301),2)

((mat ,101),1) ((sat ,201),1) ((cat ,301),1)

((sat ,101),1) ((eat ,201),1) ((pat ,301),1)

((eat ,101),1) ((fat ,301),1)

**S-S (Reducers Input):-**

**Reducer 1 Input Reducer 2 Input**

((cat ,101),[2]) ((eat ,101),[1])

((cat ,301),[1]) ((eat ,201),[1])

((mat ,101),[1]) ((fat ,301),[1])

((mat ,201),[2]) ((pat ,101),[1])

((mat ,301),[2]) ((pat ,201),[2])

((sat ,101),[1]) ((pat ,301),[1])

((sat ,201),[1])

((sat ,301),[1])

**Reducers Output:-**

**Reducer 1 Output**

(cat,[(101,2),(301,1)]) // [] is a linked list and () is a node.

(mat,[(101,1),(201,2),(301,2)])

(sat,[(101,1),(201,1),(301,1)])

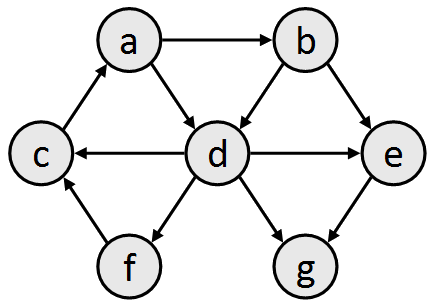
**Reducer 2 Output**

(eat,[(101,1),(201,1)])

(fat,[(301,1)])

(pat,[(101,1),(201,2),(301,1)])

Q2 Illustrate Page Rank Algorithm (Three steps only).



(2) (3) (5) (7) (8) (9) (10) (11)

(2) 0 0 0 0 0 0 0 0

(3) 0 0 0 0 1/2 0 1/2 0

(5) 0 0 0 0 0 0 0 1

(7) 0 0 0 0 1/2 0 0 1/2

(8) 0 0 0 0 0 1 0 0

(9) 0 0 0 0 0 0 0 0

(10) 0 0 0 0 0 0 0 0

(11) 1/3 0 0 0 0 1/3 1/3 0

n = 8

Init value: 1/n = 1/8 = 0.125

Initial PageRank Values , 0th Step

(2) is : (1/8)

(3) is : (1/8)

(5) is : (1/8)

(7) is : (1/8)

(8) is : (1/8)

(9) is : (1/8)

(10) is : (1/8)

(11) is : (1/8)

PageRank Values, 0th Step

(2) is : (1/3 \* 1/8) = (1/24) = 0.041666667

(5) is : (1/8) \* 0 = 0

(3) is : (1/8) \* 0 = 0

(7) is : (1/8) \* 0 = 0

(8) is : (0.5 \* 1/8) + (0.5 \* 1/8) = (1/8) = 0.125

(9) is : (1 \* 1/8) + ((1/3) \* 1/8) = (1/6) = 0.166666667

(10) is : (1/8 \* 1/2) + (1/8 \* 1/3) = (5/48) = 0.104166667

(11) is : (1/8 \* 1/2) + (1/8 \* 1) = (3/16) = 0.1875

PageRank Values, 1th Step

(2) is : (1/3 \* (3/16)) = (1/16) = 0.0625

(5) is : (0) \* 0 = 0

(3) is : (0) \* 0 = 0

(7) is : (0) \* 0 = 0

(8) is : (0.5 \* 0) + (0.5 \* 0) = (0) = 0

(9) is : (1 \* (1/8)) + ((1/3) \* (3/16)) = (3/16) = 0.1875

(10) is : (0.5 \* (0)) + ((1/3) \* (3/16)) = (5/48) = 0.0625

(11) is : (1 \* (0)) + (0.5 \* (0)) = 0

PageRank Values, 2th Step

(2) is : (1/3 \* (0)) = 0

(5) is : (0) \* 0 = 0

(3) is : (0) \* 0 = 0

(7) is : (0) \* 0 = 0

(8) is : (0.5 \* 0) + (0.5 \* 0) = 0

(9) is : (1 \* (0)) + ((1/3) \* (0)) = 0

(10) is : (0.5 \* (0)) + ((1/3) \* (0)) = 0

(11) is : =(1 \* (0)) + (0.5 \* (0)) = 0S

Q3 Apply **all the data compression algorithms (covered in class by the professor through examples)**on the following list of postings.

[(512, 15), (2080, 93), (5748, 195)]

[(512, 15) , (1568, 93) , (3668, 195)]

Byte-Aligned code

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 512 | 15 | 1568 | 93 | 3668 | 195 |
| 10 0000 0000 | 1111 | 110 0010 0000 | 101 1101 | 1110 0101 0100 | 1100 0011 |
| **0**0000100 **1**0000000 | **1**0001111 | **0**0001100 **1**0100000 | **1**1011101 | **0**0011100**1**1010100 | **0**0000001**1**1000011 |
| **0**0000100 **1**0000000 **1**0001111 **0**0001100 **1**0100000 **1**1011101 **0**0011100**1**1010100 **0**0000001**1**1000011 | | | | | |