Mapreduce, gn, cp, pagerank, fm, dgim Numericals

Big Data Analytics (University of Mumbai)

→ ONE STEP

3×3×2

PAGE :
DATE : / /

Q. Mapreduce.

$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}_{3\times3}$ $B = \begin{bmatrix} 10 \\ 11 \\ 12 \end{bmatrix}_{3\times1}$

$A_{ij} = \begin{bmatrix} 3 & 4 \\ 7 & 2 \\ 5 & 9 \end{bmatrix}_{3\times2}$ $B_{jk} = \begin{bmatrix} 3 & 1 & 5 \\ 6 & 9 & 7 \end{bmatrix}_{2\times3}$

$i = 3, \; j = 2, \; k = 3$

① Map.

(Matrix A)		(Matrix B)		
(i, k)	(A, j, A_{ij})	(i, k),	(B, j, B_{jk})	
(1,1)	(A, 1, 3)	(1,1)	(B, 1, 2)	3
(1,2)	(A, 1, 3)	(1,2)	(B, 1, 3)	1
(1,3)	(A, 1, 3)	(1,3)	(B, 1, 3)	5
(1,1)(2,1)	(A, 2, 4)	(1,1)	(B, 2, 1)	6
(1,2)(2,2)	(A, 2, 4)	(1,2)	(B, 2, 1)	9
(1,3)(2,3)	(A, 2, 4)	(1,3)	(B, 2, 1)	7
(2,1)(3,1)	(A, 1, 7)	(2,1)	(B, 3, 8)	3
(2,2)(3,2)	(A, 1, 7)	(2,2)	(B, 3, 8)	1
(2,3)(3,3).	(A, 1, 7)	(2,3)	(B, 3, 8)	5
(2,1)	(A, 2, 2)	(2,1)	(B, 1, 6)	6
(3,2)	(A, 2, 2)	(2,2)	(B, 1, 6)	9
(3,3)	(A, 2, 2)	(2,3)	(B, 1, 6)	7
(3,1)	(A, 1, 5)	(3,1)	(B, 2, 9)	3
(3,2)	(A, 1, 5)	(3,2)	(B, 2, 9)	1
(3,3)	(A, 1, 5)	(3,3)	(B, 2, 9)	5
(3,1)	(A, 2, 9)	(3,1)	(B, 3, 7)	6
(3,2)	(A, 2, 9)	(3,2)	(B, 3, 7)	9
(3,3)	(A, 2, 9)	(3,3)	(B, 3, 7)	7

PAGE :
DATE : / /

~~Reducer~~ Shuffle

$(X, j, x_{ij}/x_{jk})$

(i, k)

(1,1) (A,1,3), (A,2,4)
(B,1,3) (B,2,1)

(1,2) (A,1,3) (A,2,4)
(B,1,1) (B,2,9)

(1,3) (A,1,3) (A,2,4)
(B,1,5) (B,2,7)

(2,1) (A,1,7) (A,2,2)
(B,3,3) (B,1,6)

(2,2) (A,1,7) (A,2,2)
(B,3,1) (B,1,9)

(2,3) (A,1,7) (A,2,2)
(B,3,5) (B,1,7)

(3,1) (A,1,5) (A,2,9).
(B,2,3) (B,3,6)

(2,2) (A,1,5) (A,2,9)
(B,2,1) (B,3,9)

(3,3) (A,1,5) (A,2,9)
(B,2,5) (B,3,7).

(2) ~~Reducer~~

~~(i, j, k)~~

(3) Reducer - for matching (i, k) diff x, vertically multiply the values. & add the product. horizontally.

(1,1) (3×3 + 4×1) = 13
(1,2) (3×1 + 4×9) = 39
(1,3) (3×5 + 4×7) = 43
(2,1) (7×3 + 2×6) = 33
(2,2) (7×1 + 2×9) = 25
(2,3) (7×5 + 2×7) = 49

PAGE :
DATE : / /

(3,1) (5×3 + 9×6) = 87
(3,2) (5×1 + 9×9) = 86
(3,3) (5×5 + 9×7) = 88

$$\text{Product} = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & \cancel{13} & \cancel{39} & \cancel{43} \\ 2 & \cancel{33} & \cancel{25} & \cancel{49} \\ 3 & \cancel{87} & \cancel{86} & \cancel{88} \end{array} \qquad \begin{bmatrix} 13 & 33 & 87 \\ 39 & 25 & 86 \\ 43 & 49 & 88 \end{bmatrix}$$

* TWO STEP

$$M = \begin{array}{c|cc} & 1 & 2 \\ \hline 1 & 2 & 3 \\ 2 & 4 & 9 \end{array}_{\,i \times j} \qquad N = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 5 & 6 & 8 \\ 2 & 2 & 4 & 7 \end{array}_{\,j \times k}$$

i = 2, j = 2, k = 3.

Step 1 – Mapper. { j (M, i, M_{ij}) ; j (N, k, N_{jk}) }

j (M, i, M_{ij})	j (N, k, N_{jk})
1 (M, 1, 2)	1 (N, 1, 5)
1 (M, 2, 4)	1 (N, 2, 6)
2 (M, 1, 3)	1 (N, 3, 8)
2 (M, 2, 9)	2 (N, 1, 2)
	2 (N, 2, 4)
	2 (N, 3, 7)

Reducer j (i, k, $M_{ij} \times N_{jk}$)

1 (1, 1, 2×5) ⇒ 1 (1, 1, 10)
1 (2, 1, 4×5) ⇒ 1 (2, 1, 20)
1 (1, 2, 2×6) ⇒ 1 (1, 2, 12)
1 (2, 2, 4×6) ⇒ 1 (2, 2, 24)
1 (1, 3, 2×8) ⇒ 1 (1, 3, 16)
1 (2, 3, 4×8) ⇒ 1 (2, 3, 32)

PAGE :
DATE : / /

2 (1,1 , 3×2) ⇒ 2(1, 1, 6)
2 (2,1, 9×2) ⇒ 2(2, 2, 18)
2 (1,2 , 3×4) ⇒ 2(1, 2, 12)
2 (2,2, 9×4) ⇒ 2(2, 2, 36)
2 (1,3, 3×7) ⇒ 2(1, 3, 21)
2 (2,3, 9×7) ⇒ 2(1, 2, 63)

Step 2 - Mapper.

(j [(i₁, k₁, v₁), (i₂, k₂, v₂) . . . . .])

(1 [(1,1,10), (1,2,12), (1,3,16),
(2,1,20), (2,2,24), (2,3,32)])

(2 [(1,1 6), (1,2,12), (1,3,21),
(2,1,18), (2,2,36), (2,3,63)].

Reducer

For each (i, k), add the corresponding values.

[(1,1), 10+6] [(2,1), 20+18]
[(1,2), 12+12] [(2,2), 24+36]
[(1,3), 16+21] [(2,3), 32+63]

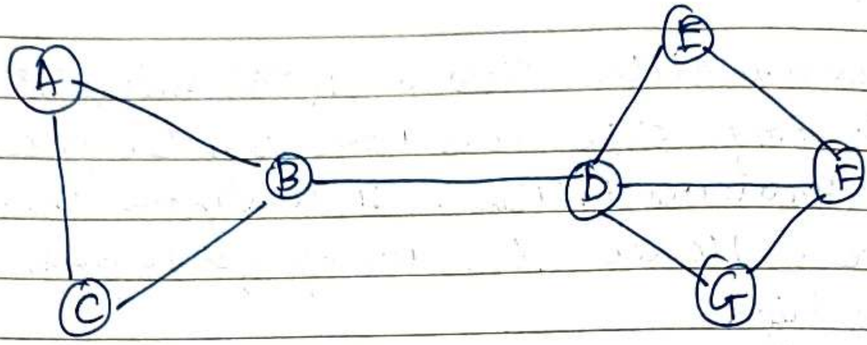
$$P = \begin{bmatrix} 16 & 24 & 37 \\ 38 & 60 & 95 \end{bmatrix}$$

DEF

DGF

PAGE :
DATE : / /

Q. GN algorithm.



~~Shortest paths.~~ **EB**

Path.

AB ⇒ AB, AD, AE, AG, AF, AC ⇒ 5

AC ⇒ AC ⇒ 1

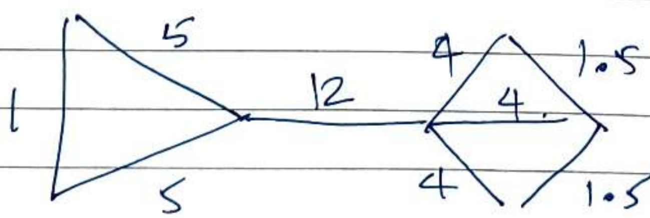
BC ⇒ AB, AD, AE, AG, AF, AC ⇒ 5

BD ⇒ { BD, BE, BF, BG, AD, AE, AF, AG, CD, CE, CF, CG } ⇒ 4×3 = 12

DE ⇒ DE, CE, BE, AE } = DG ⇒ 4

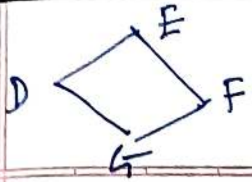
DF ⇒ DF, BF, AF, CF ⇒ 4.

EF ⇒ { EF, EG $\frac{1}{2}$ } ⇒ ED ⇒ 1.5



Breaking the edge with highest EB. {BD}.

{  } final communities.



$DF \rightarrow DE,\ DF^{1/2}$ | $EF \rightarrow EF,\ EG^{1/2}$

$DG \rightarrow DG,\ DF^{1/2}$ | $FG \rightarrow FG,\ EG^{1/2}$

EB. i.e,

Algo ⇒ ① calculate the total no. of shortest paths from some pt α to some pt β that pass through edge E.

② Break the edge with highest EB factor.

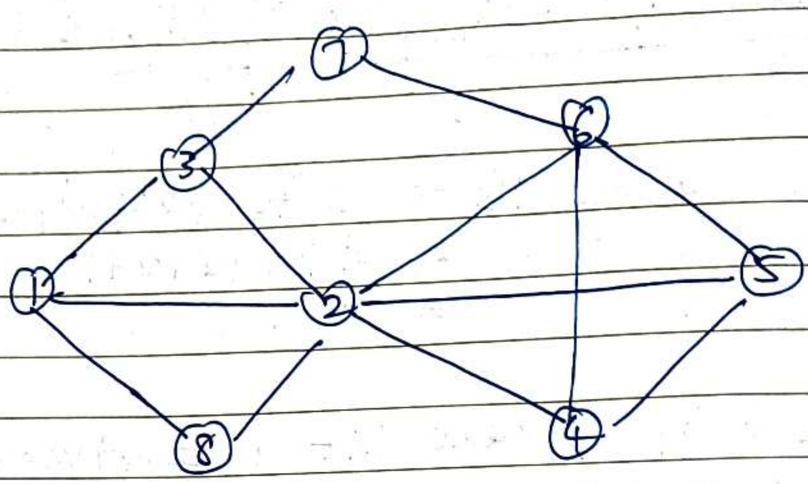
③ Repeat till required no. of comm is obtained or all edges in a comm have the same EB factor.

⇒ Points.

- hierarchical clustering model
- used for ① comm. detection
  ② measure edge betweenness
- $EB_E = \left\{ \begin{array}{l} \text{no of shortest paths from some node} \\ \text{X \& some ~~other~~ node Y that passes} \\ \text{through edge E} \end{array} \right\}$.
- if there are ~~two~~ n shortest paths between X & Y, then ~~both~~ all Edges ~~E_1 & E_2~~ $E_1, E_2 \ldots E_n$ will have $(1/n)$ weight.
- complexity ⇒ ① calculation of edge betn $= O(EN)$
  ② algorithm $= O(E^{2N})$

PAGE :
DATE : / /

Q. Clique percolation method.



Identifying 3 cliques. ⇒ 6 3-cliques.

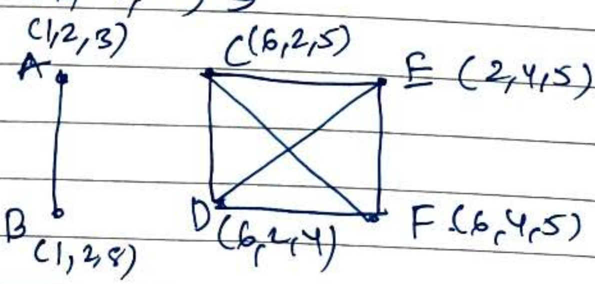
A: (1, 2, 3)
B: (1, 2, 8)
C: (6, 2, 5)
D: (6, 2, 4)
E: (2, 4, 5)
F: (6, 4, 5)

Combining cliques into communities.
(cliques should have 2 nodes in common).

A: (1, 2, 3)
B: (1, 2, 8) } ⟶ C_1 : (1, 2, 3, 8)

C: (6, 2, 5)
D: (6, 2, 4)
E: (2, 4, 5)
F: (6, 4, 5) } ⟶ C_2 : (2, 4, 5, 6)

⟶ ~~C3: (2, 4, 5,~~

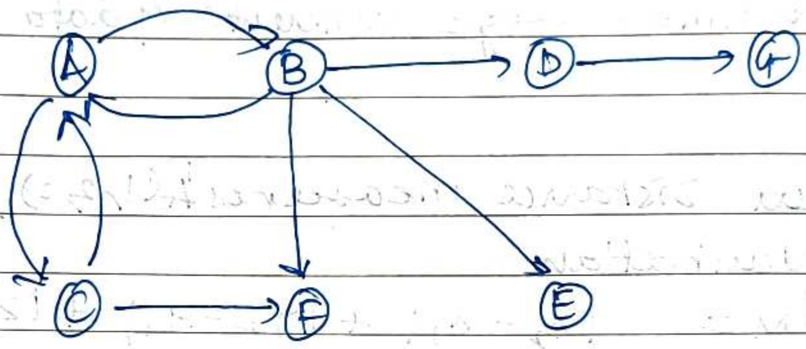


PAGE :
DATE : / /

Q. Pagerank Algorithm

Purpose is to rank the webpages based on the no of incoming & outgoing links in that page. A more important page will have larger no. of such links.

eg.



In this case F, E & G are deadends.

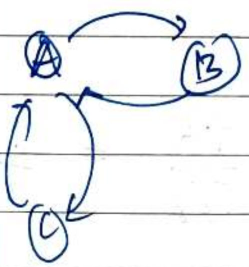
D becomes deadend.

Transition Matrix

$$M_T = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\ \begin{bmatrix} 0 & 1/4 & 1/2 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$\text{new } M_T = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 1 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \end{bmatrix} \end{array}, \quad V_1 = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

initial rank



PAGE :
DATE : / /

To handle spider trap (group ~~of~~ of pages that have no outgoing links. we introduce teleport factor into the eqn. (β=0.85). we assume that the web server will visit a page with α prob. & start ~~new~~ fresh browsing with $(1-\beta)$ prob. i.e. teleports to some random page with $(1-\beta)$ prob.

new page rank ⇒

$$v' = \beta M v + (1-\beta) \cdot e/n$$

iteration 0.

$$v_1' = 0.85 \begin{bmatrix} 0 & 1 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} + 0.15 \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}.$$

$$v_1' = \frac{1}{120} \cancel{74} \begin{bmatrix} 74 \\ 23 \\ 23 \end{bmatrix}$$

iteration 1

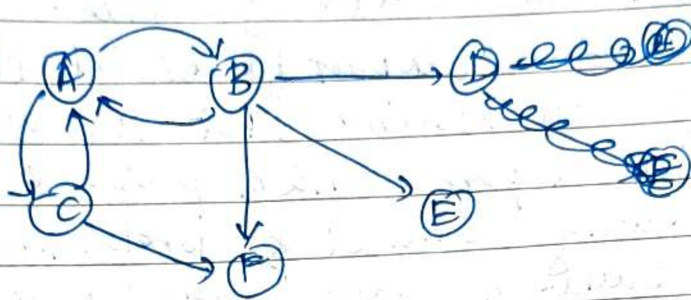
$$v_2' = \frac{0.85}{120} \begin{bmatrix} 0 & 1 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 74 \\ 23 \\ 23 \end{bmatrix} + 0.15 \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}.$$

$$= \frac{1}{2400} \begin{bmatrix} 902 \\ 749 \\ 749 \end{bmatrix}.$$

$$PR_A = \frac{902}{2400}, \quad PR_B = \frac{749}{2400}, \quad PR_C = \frac{749}{2400}.$$

adding deadends D, E, F, ~~G~~

PR

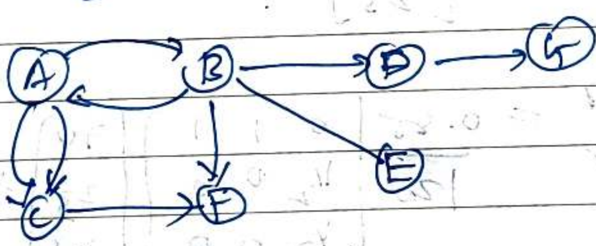


$$PR_D = \frac{PR_B}{4} =$$

$$PR_E = \frac{PR_B}{4}$$

$$PR_F = \frac{PR_B}{4}.$$

adding G



$$PR_G = PR_D.$$

PAGE:
DATE: / /

Q. FM Algorithm.

- Used to count unique nos in a data stream.
- use hashing ~~them~~ functions
- time complexity is $O(n)$
  memory req - $O(\log(m))$

Algo -

- make an array of stream elements.
- take a hash function $\{ h(x) = (5x + 1) \% 4$
- find hash values ~~of~~ for each element
- convert ~~hash val~~ binary of each hash value
- count no of zeros in each bin. value
- ~~max of~~ find max no of trailing zeros.
- no. of unique values in stream $t \approx 2^r$ (nearly accurate)

eg. $S = 1, 3, 2, 1, 2, 3, 4, 3, 1, 2, 3, 1$

x	1	3	2	1	2	3	4	3	1	2	3	1
$h(x)$	2	0	3	2	3	0	1	0	2	3	0	2
$B(x)$	010	000	011	000	011	000	001	000	010	011	100	010
trailing zeros.	1	2	0	2	0	2	0	2	1	0	2	1

$r = \max(\text{trailing zeros}) = 2.$

no of unique values $= 2^r = 2^2 = \underline{4}$ $\{1, 2, 3, 4\}$

DGIM

1 0 1 1 0 1 0 1 0 1 1 0 1 1 1 0 0 1 1

3 6 9 12 15 18 19.

t=100 1 0 1 1 0 1 0 1 0 1 1 0 1 1 1 0 0 1 1

⇒ 101101 0 101101 11 00 1 1 (19)

t=101 1 arr. 101101 0 101101 11 00 1 1 1

⇒ 101101 0 101101 11 00 11 1 (20)

t=102 0 arr. 101101 0 101101 11 00 11 1 0 (21)

t=103 1 arr. 101101 0 101101 11 00 11 1 0 1 (22)

t=104 1 arr. 101101 0 101101 11 00 11 1 0 1 1

⇒ 101101 0 101101 110011 101 1 (23)

t=105 1 arr 101101 0 101101 110011 101 1 1 (24)

WINDOW

⇒ 101101 0101101 110011 101 1 1

2^3 2^2 2^1 2^0 2^0

largest bucket is partly into the window.

calculated

total no of 1's $= \frac{1}{2}(2^3) + 2^2 + 2^1 + 2^0 + 2^0$.

$= \frac{1}{2}(8) + 4 + 2 + 1 + 1$

total no of 1's. $= 12$

Actual no of 1's $= 16$

% accuracy $= \frac{\Delta(1's)}{AV.} \times 100 = \frac{4}{16} \times 100$

$= 25\%$.

* DGIM

101011000101110110010110

N = 24 (window size).

[1] 0 [1] 0 [1] [1] 0 0 0 [1] 0 [1] [1] [1] 0 [1] [1] 0 0 [1] 0 [1] [1] 0

[101011] 000 [10111] 0 [11] 00 [101] [1] 0 ← 0 1 1

Timestamps → 87 ...... 92 ..... 95 ...... 98 ... 100 ← 101 102 103

[101011] 000 [10111] 0 [11] 00 [101] [1] 0 0 ← 1 1 1

87 92 95 98 100 101 102 103 104.

[101011] 000 [10111] 0 [11] 00 [101] [1] 0 0 [1] ← 1 1

87 92 95 98 100 101 102 103 104

[101011] 000 [10111] 0 [11] 00 [101] [1] 00 [1] [1] ← 1

87 92 95 98 100 101 102 103 104.

87 92 95 102

[10101100010111] 0 [1100101] [1001] [1] [1]

103 104

2^3 2^2 2^1 2^0 2^0

if current ts − leftmost bucket ts of window < N, continue

Eg. 103 − 87 = 16 < 24 ∴ continue.

if greater or equal, stop.

How many 1's are there in the last 20 bits?

$$= \frac{2^3}{2} + 2^2 + 2^1 + 2^0 + 2^0$$

$$= 4 + 4 + 2 + 1 + 1 = \underline{12}$$