Thi Hong Yen Truong

ID: 614745

**W2D2**

**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]

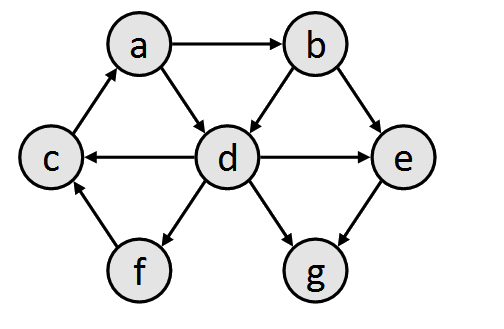
Input Split 3 (Doc ID 301) [sat mat cat pat fat mat]

Assume three Mappers and two Reducers.

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

|  |  |  |  |
| --- | --- | --- | --- |
| Input Split 1 (Doc ID 101) | Input Split 2 (Doc ID 201) | | Input Split 3 (Doc ID 301) |
| [cat pat mat sat cat eat] | [pat mat sat pat mat eat] | | [sat mat cat pat fat mat] |
| Mapper 1 output | Mapper 2 output | | Mapper 3 output |
| ((cat, 101), 2)  ((pat, 101), 1)  ((mat, 101), 1)  ((sat, 101), 1)  ((eat, 101), 1) | ((pat, 201), 2)  ((mat, 201), 2)  ((sat, 201), 1)  ((eat, 201), 1) | | ((sat, 301), 1)  ((mat, 301), 2)  ((cat, 301), 1)  ((pat, 301), 1)  ((fat, 301), 1) |
| Shuffle and Sort | | | |
| Reducer 1 input | | Reducer 2 input | |
| ((cat, 101), [2])  ((cat, 301), [2])  ((mat, 101), [1])  ((mat, 201), [2])  ((mat, 301), [2])  ((sat, 101), [1])  ((sat, 201), [1])  ((sat, 301), [1]) | | ((eat, 101), [1])  ((eat, 201), [1])  ((fat, 301), [1])  ((pat, 101), [1])  ((pat, 201), [2])  ((pat, 301), [1]) | |
| Reducer 1 output | | Reducer 2 output | |
| (cat, [(101, 2), (301, 2)])  (mat, [(101, 1), (201, 2), (301, 2)])  (sat, [(101, 1), (201, 1), (301, 1)]) | | (eat, [(101, 1), (201, 1)])  (fat, [(301, 1)])  (pat, [(101, 1), (201, 2), (301, 1)]) | |

**Q2. Illustrate Page Rank Algorithm (Three steps only).**



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Steps | a | b | c | d | e | f | g |
| 0 | 1/7 | 1/7 | 1/7 | 1/7 | 1/7 | 1/7 | 1/7 |
| 1 | 1/7 | 1/14 | 5/28 | 1/7 | 3/28 | 1/28 | 5/28 |
| 2 | 5/28 | 1/14 | 1/14 | 3/28 | 1/14 | 1/28 | 1/7 |

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

[(512, 15), (2080, 93), (5748, 195), (7080, 255] => Total is 24 bytes

**Step: Common to all compression schemes (d-Gaps):**

[(512, 15), (1568, 93), (3668, 195), (1332, 255)]

* 512 15 1568 93 3668 195 1332 255

**Step: Byte-aligned codes:**

512 = 1000000000

|  |  |
| --- | --- |
| 0000 0100 | 1000 0000 |

15 = 1111

|  |
| --- |
| 1000 1111 |

1568 = 1100010000

|  |  |
| --- | --- |
| 0000 0110 | 1001 0000 |

93 = 1011101

|  |
| --- |
| 1101 1101 |

3668 = 111001010100

|  |  |
| --- | --- |
| 0001 1100 | 0101 0100 |

195 = 11000011

|  |  |
| --- | --- |
| 0000 0001 | 1100 0011 |

1332 = 10100110100

|  |  |
| --- | --- |
| 0000 1010 | 1011 0100 |

255 = 11111111

|  |  |
| --- | --- |
| 0000 0001 | 1111 1111 |

Compressed -> Total is 14 bytes