Analyzing a Graph with Hadoop/Java

(a)

Step 1: Set up development environment for MapReduce and load data to Hadoop Distributed File System.

Step 2: Split each line of graph data in tsv file with a Tab and save to a String array. As for the given example:

src	tgt	weight
100	10	3
110	10	3
200	10	1
150	130	30
110	130	67
10	101	15

After splitting, the first line is written as array[0] = 100, array[1] = 10, array[2] = 3.

Step 3: Do mapper process. Set the <key, value> pair as <tgt, weight>. For the given example above, we get the following map:
<10, 3>
<10, 3>
<10, 1>
<130, 30>
<130, 67>
<101, 15>

Step 4: Do reducer process. First we combine all pairs according to key value.
<10, <3,3,1>>
<130, <30, 67>>
<101, 15>

Then iterate each value for one key, and find the minimum value among them.
<10, 1>
<130, 30>
<101, 15>

Step 5: Adjust the output format and save all <key, value> pairs to file system.

(b)

Step 1: Set up development environment for MapReduce and load data to Hadoop Distributed File System.

Step 2: Use secondary sorting strategy. Since we want student name to always precede department name, we tag key "Student" with "1" and key "Department" with "2" to determine order of the values. For the given example,

Student	Alice	1234
Student	Bob	1234
Department	1123	CSE
Department	1234	CS
Student	Carol	1123

We get,

Student	Alice	1234	1
Student	Bob	1234	1
Department	1123	CSE	2
Department	1234	CS	2
Student	Carol	1123	1

Step 3: Do mapper process. Use Department ID as key, all remaining string as value. For the given example above, we get the following map:

<1234, <Student, Alice, 1>>
<1234, <Student, Bob, 1>>
<1123, <Department, CSE, 2>>
<1234, <Department, CS, 2>>
<1123, <Student, Carol, 1>>

Step 4: Do reducer process with secondary sorting strategy. First we combine all pairs according to key value as well as tag value.

<1234, <<Student, Alice, 1>, <Student, Bob, 1>, <Department, CS, 2>>>
<1123, <<Student, Carol, 1>, <Department, CSE, 2>>>

Then iterate each value for one key, for those tag == 1, append the student value array with department value array.

<1234, <Student, Alice, 1, Department, CS, 2>>
<1234, <Student, Bob, 1, Department, CS, 2>>
<1123, <Student, Carol, 1, Department, CSE, 2>>

Step 5: Sort Department_ID in ascending order, and adjust the format of data to specific schema, we get,

<1234, <Alice, CS>>
<1234, <Bob, CS>>
<1123, <Carol, CSE>>