The Algorithm for map reduce code makes the following assumption -

1. The input file which is a Text Sequence file is assumed to be of such a format
that when it is input to the SequenceFileInputFormat.class fileformat, the mapper
automatically gets the key and value present in the file as arguments to the map function.
2. The input file can also be converted to a text file from Sequence file using the command
"hadoop fs -text <sequenceFile>"
which would be an extra overhead and hence I did not opt this path.
3. The optimization part is not considered in the mapper and reducer but indicated as appropriate in the description.

Algorithm logic-

Input for the Mapper -
Key and Value are as read by the Sequence file reader internal to RecordReader.
Output for the Mapper -
Key is in TextFormat and corresponds to all the text entities present in keys and values of Input.
Value is in TextFormat and indicates if the text key is present in keySet(value.get() is 1) or ValueSet(value.get() is 0).

The mapper takes the keys and values follows the following logic -
1. The key is emitted with a value of 1.
2. i. The value in values list is converted to String as it is a Text object.
ii. The string is then split with tab as delimiter and each of the value, if valid,
is emitted with a value of 0.
3. There is a possible optimization here where a combiner can be employed to check
and count the emitted values local to the mapper input split.

The reducer takes the values and keys emitted by the mapper and follows the following logic -
1. The key in the reducer is taken as it is.
2. The value in the iterable set of IntWritables is converted to primitive int and summed up. Now here there are two possibilities.
i. Either the key text did not appear in the keySet at all. Its total sum would be 0.
If the total sum is 0, then we know for sure that the key text is dangling.
ii. The key text did appear in the keySet and therefore the corresponding value 1 would be added
to the total sum which makes the total sum greater than 0. Hence we will know that the key text is not dangling.

Ex.
Input split 1
Key = a
Values = b c a s

Input Split 2
Key = d
Values = e a k

Input Split 3
Key = s
Values = a k

Input Split 4
Key = b
Values = c

Mapper output 1- < <a, [1, 0]>, <b, [0]>, <c, [0]>, <s, [0]>>
Mapper output 2- < <d, [1]>, <e, [0]>, <a, [0]>, <k, [0]>>
Mapper output 3- < <s, [1]>, <a, [0]>, <k, [0]>>
Mapper output 4- <<b, [1]>, <c, [0]>>

Reducer Input -
< <a, [1, 0, 0, 0]>, <b, [1, 0]>, <c, [0, 0]>, <d, [1]>, <e, [0]>, <k, [0,0]>, <s, [1]>>
As it can be observed - c, e and k sum to 0 and others - a, b, d, s sum to 1.
Hence,
Reducer Output -
< <c, 0>, <e, 0>, <k, 0>>