Solution Q3a: Illustrate algorithm 3.8.

<table>
<tr><th>INPUT</th><th>Input Split-1</th><th>Input Split-2</th></tr>
<tr><td>Mapper Input</td><td>
<table>
<tr><td>cat mat rat cat</td></tr>
<tr><td>cat bat cat pat</td></tr>
<tr><td>cat bat rat bat</td></tr>
</table>
Neighbours:

N(cat) = {mat,rat}
N(mat) ={rat,cat}
N(rat) = {cat}
N(cat) = {}

N(cat) = {bat}
N(bat) = {cat,pat}
N(cat) = {pat}
N(pat) = {}

N(cat) = {bat,rat,bat}
N(bat) = {rat}
N(rat) = {bat}
N(bat) = {}
</td><td>
<table>
<tr><td>cat rat bat rat</td></tr>
<tr><td>bat mat pat bat</td></tr>
<tr><td>pat cat bat mat</td></tr>
</table>
Neighbours:

N(cat) = {rat,bat,rat}
N(rat) = {bat}
N(bat) = {rat}
N(rat) = {}

N(bat) = {mat,pat}
N(mat) = {pat,bat}
N(pat) = {bat}
N(bat) = {}

N(pat) = {cat,bat,mat}
N(cat) = {bat,mat}
N(bat) = {mat}
N(mat) = {}
</td></tr>
<tr><td>MAP</td><td>Mapper-1</td><td>Mapper-2</td></tr>
<tr><td>Mapper Output</td><td>
((cat,mat),1)
((cat,rat),1)
((mat,rat),1)
((mat,cat),1)
((rat,cat),1)

((cat,bat),1)
((bat,cat),1)
((bat,pat),1)
((cat,pat),1)

((cat,bat),1)
((cat,rat),1)
((cat,bat),1)
((bat,rat),1)
((rat,bat),1)
</td><td>
((cat,rat),1)
((cat,bat),1)
((cat,rat),1)
((rat,bat),1)
((bat,rat),1)

((bat,mat),1)
((bat,pat),1)
((mat,pat),1)
((mat,bat),1)
((pat,bat),1)

((pat,cat),1)
((pat,bat),1)
((pat,mat),1)
((cat,bat),1)
((cat,mat),1)
((bat,mat),1)
</td></tr>
</table>

PARTITION	**(a-j)**	**(k-z)**
	((cat,mat),1) ((cat,rat),1) ((cat,bat),1) ((bat,cat),1) ((bat,pat),1) ((cat,pat),1) ((cat,bat),1) ((cat,rat),1) ((cat,bat),1) ((bat,rat),1) ((cat,rat),1) ((cat,bat),1) ((cat,rat),1) ((bat,rat),1) ((bat,mat),1) ((bat,pat),1) ((cat,bat),1) ((cat,mat),1) ((bat,mat),1)	((rat,bat),1) ((mat,pat),1) ((mat,bat),1) ((pat,bat),1) ((pat,cat),1) ((pat,bat),1) ((pat,mat),1) ((mat,rat),1) ((mat,cat),1) ((rat,cat),1) ((rat,bat),1)
SORT & COMBINE		

Reducer Input
 	
Sorting rule:
 	
class Pair implements Comparable<Pair> {
String a, b;
int compareTo(Pair p) {
int k = a.compareTo(p.a)
if(k==0) k=b.compareTo(p.b)
return k;
}
}
	((bat,cat),[1])
	((bat,mat),[1,1])
	((bat,pat),[1,1])
	((bat,rat),[1,1])
	
	((cat,bat),[1,1,1,1,1])
	((cat,mat),[1,1])
	((cat,pat),[1])
	((cat,rat),[1,1,1,1])
		((mat,bat),[1])
		((mat,cat),[1])
		((mat,pat),[1])
		((mat,rat),[1])
		
		((pat,bat),[1,1])
		((pat,cat),[1])
		((pat,mat),[1])
		
		((rat,bat),[1,1])
		((rat,cat),[1])

REDUCE	**Reducer-1**	**Reducer-2**
Reducer Output	((bat,cat), 1) ((bat,mat), 2) ((bat,pat), 2) ((bat,rat), 2) ((cat,bat), 5) ((cat,mat), 2) ((cat,pat), 1) ((cat,rat), 4)	((mat,bat), 1) ((mat,cat), 1) ((mat,pat), 1) ((mat,rat), 1) ((pat,bat), 2) ((pat,cat), 1) ((pat,mat), 1) ((rat,bat), 2) ((rat,cat), 1)