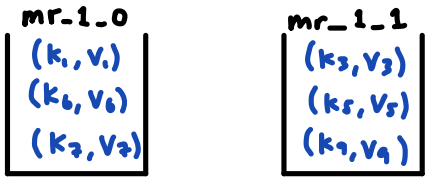
map task 0 (p.txt)

output of map function: n_{out} (k,v) pairs

Let $n_{out} = 5$.

map task 1 (q.txt)

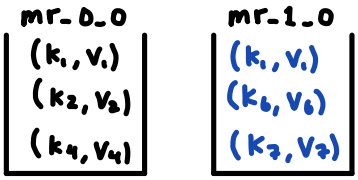


Example: p.txt contains "apple" 5 times. We're running WC.

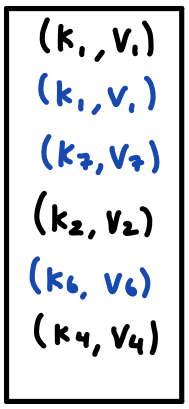
So (k_1, v_1) = ("apple", 5). q.txt also contains apple, but 3 times.

So (k_1, v_1) = ("apple", 3). Note that the k1 from map task 0 is the same as the one from map task 0.

Worker 1 now works on reduce task 0.



Concat all 6 (k,v) pairs and sort.



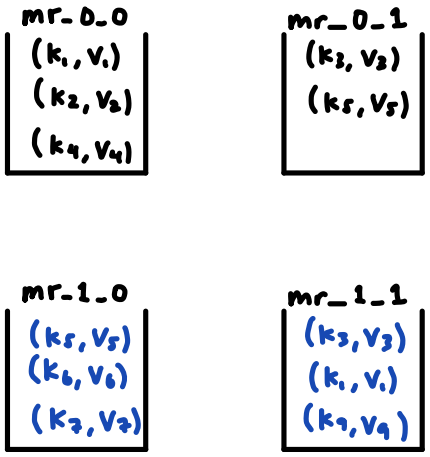
Reduce k_1: Take in (k_1, v_1) and (k_1, v_1) and produce (k_1, v_1).

Reduce k_7: Take in (k_7, v_7) and produce (k_7, v_7).

etc.

Note that the following scenario where a reduce task doesn't "fully aggregate" all the vals for a key is impossible.

Ex:



Here, reduce task 0 won't be aware of (k_1, v_1).

This is impossible b/c the hash function and modulo will put (k_1, V_1) and (k_1, V_1) into the "same bucket" (i.e. if (k_1, V_1) goes into mr_0_i, (k_1, V_1) goes into mr_1_i).