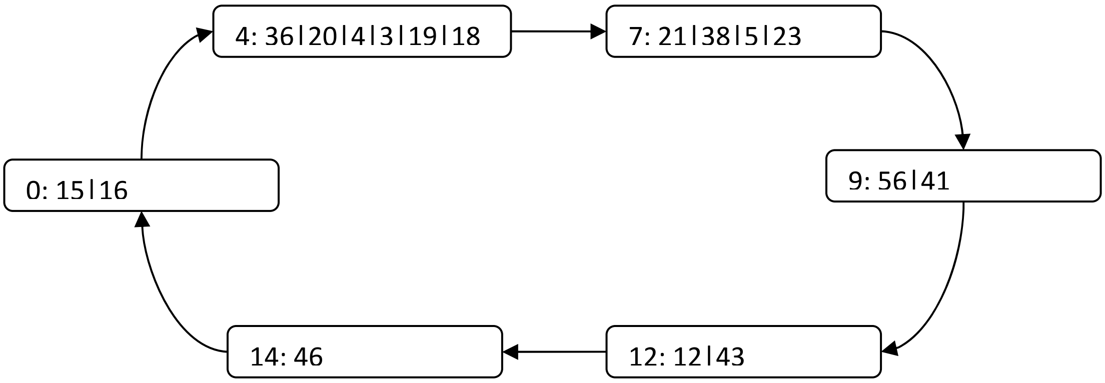
# Consistent hash

Let’s assume D=16, the hash function is simply the module of the IP address or the key, and suppose the current state of the consistent hash is (position_in_the_ring: key|key|…):

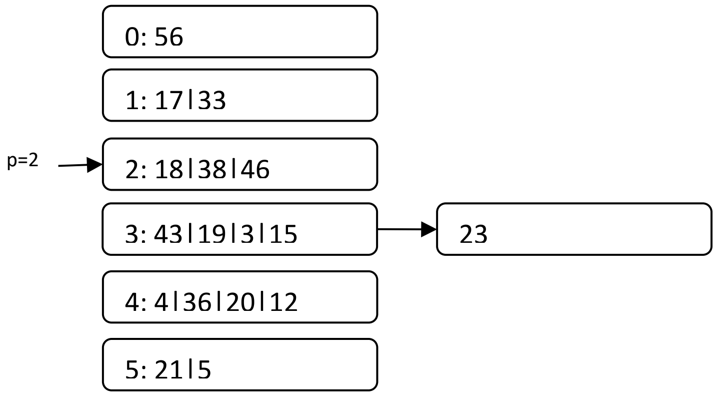


What happens when we insert objects “30” and “58”? Draw the result.

What happens in the structure when we register a new server with IP address “37”? Draw the result.

# Linear hash

Let’s suppose the hash function is simply the module of the key, the capacity of a bucket is only four entries, and current state of the linear hash is (bucketID: key|key|…):



What happens in the structure when we insert keys “14”, “27”, “37”, and “44”? Draw the result.

# LSM Tree

Let’s suppose that, we have reached the threshold to consider the MemStore is full, and it contains four entries with format [key, value, timestamp] needing ten characters each. The content of the different structures is:

MemStore: [1,v,t50], [15,v, t49], [17,v,t47], [29,v,t48]

Commit Log: [17,v,t47], [29,v,t48], [15,v, t49], [1,v,t50]

SSTable: [13,v,t23], [25,v,t17], [35,v,t40], [59,v,t38]

Index: [13,0], [25,30], [35,60], [59,90]

Assuming that the minimum size of an SSTable is 120 characters and on having two SSTables a minor compactation is automatically triggered, explicit the content of all structures once the compactation is done:

MemStore: ……………………………………………………………………………………………………………………….

Commit Log: ………………………………………………………………………………………………………………………

SSTable: ………………………………………………………………………………………………………………………….

Index: …………………………………………………………………………………………………………………………….