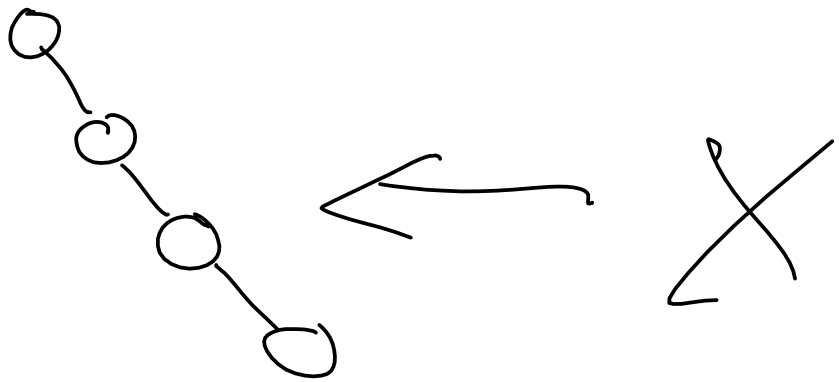
Dictionary ADT mapping keys to values

		BST	AVL	
	put(K key, V value)	O(n)	O(log n)	O(1)
V	get(K key)	O(n)	O(log n)	O(1)
V	remove(K key)	O(n)	O(log n)	O(1)
	size()	O(1)	O(1)	O(1)
	contains(K key)	O(n)	O(log n)	O(1)



$[0, 1000000)$ $[0, 10)$

$[-5000, 5000]$

K = strings

"A" "B" ... "J"
0 1 9

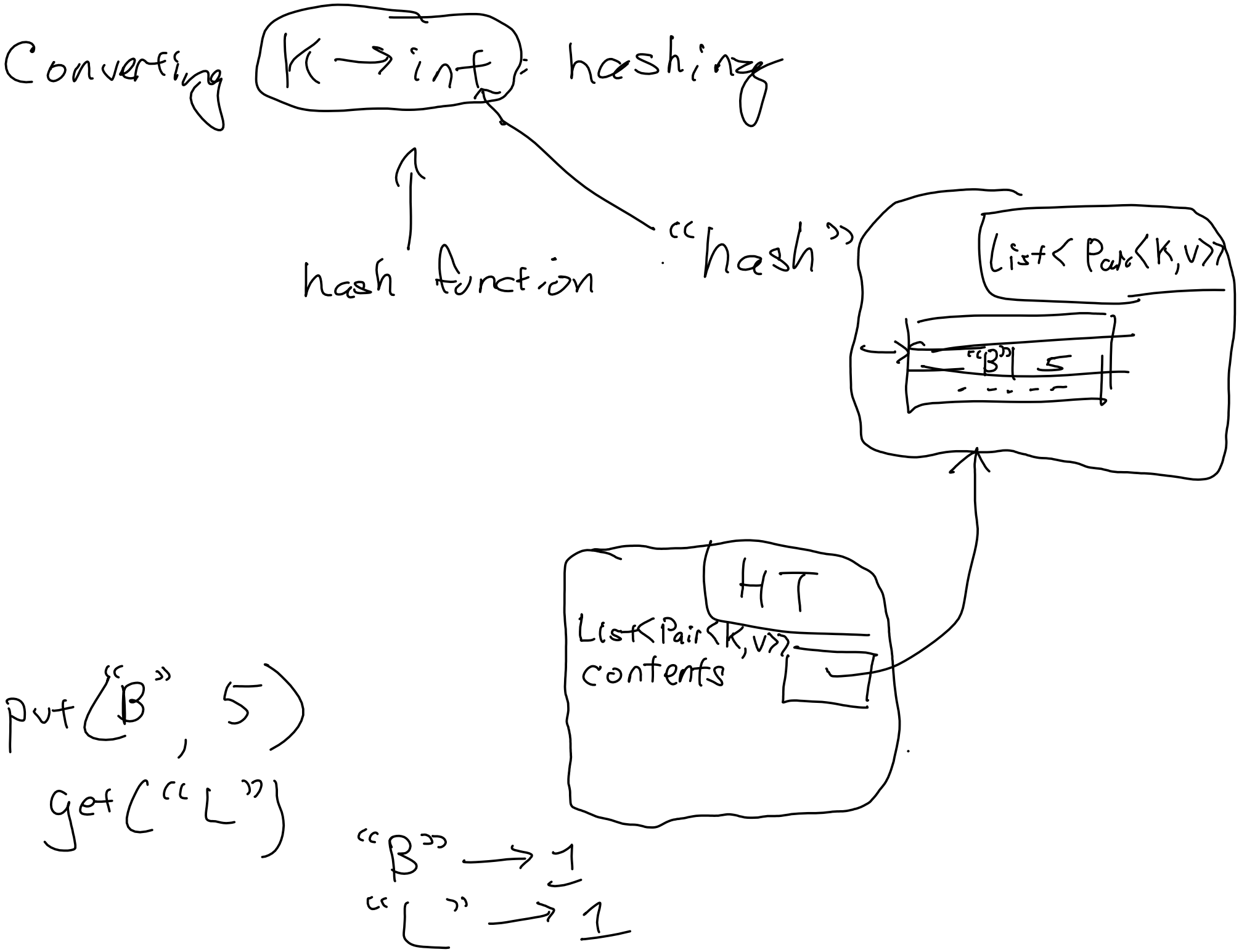
"C"
"M"
2

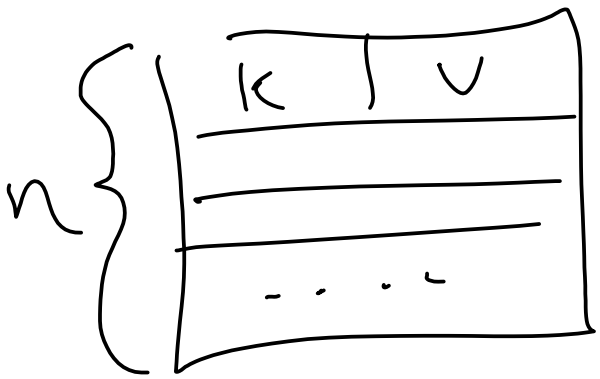
AB
01000001, 01000010

"AA"
26
6

converting K to an int

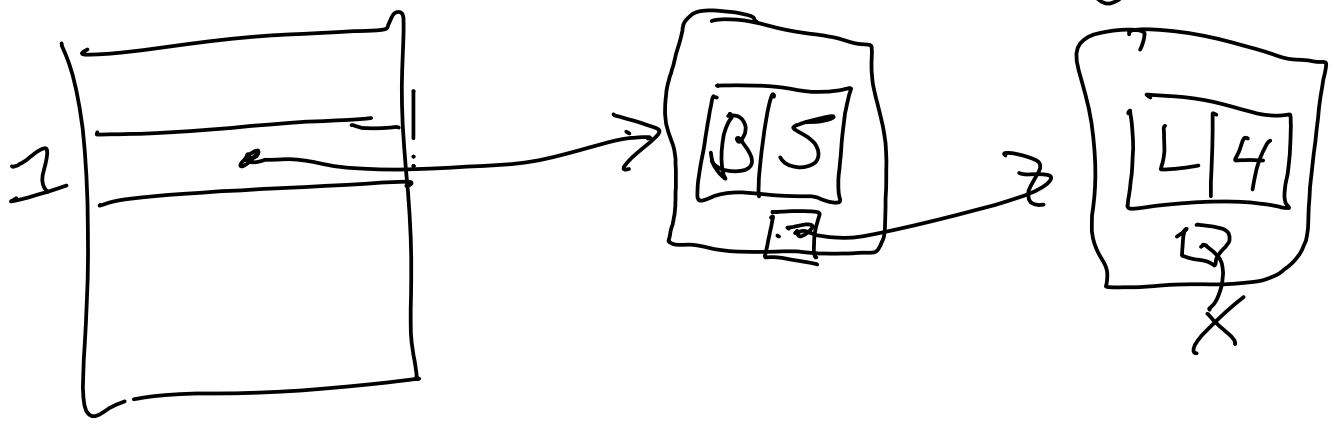
"hashing"


Converting
K → int: hashing
hash function
"hash"
List<Pair<K,V>>
"B" 5
HT
List<Pair<K,V>>
contents
put("B", 5)
get("L")
"B" → 1
"L" → 1



$h(K) \Rightarrow i$

Chaining

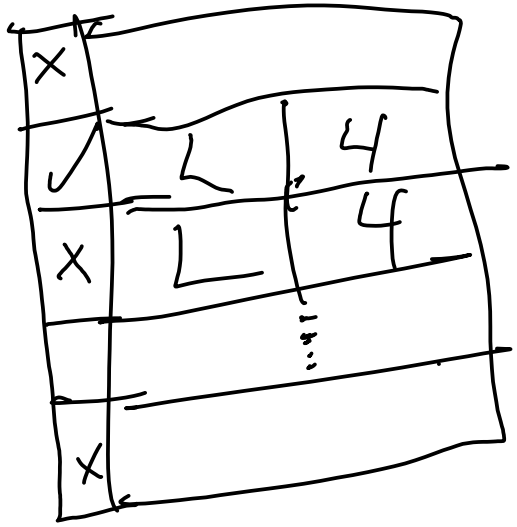


"B" ↦ 1

"L" ↦ 1

put("B", 5)

put("L", 4)



10

put("B", 5)

put("L", 4)

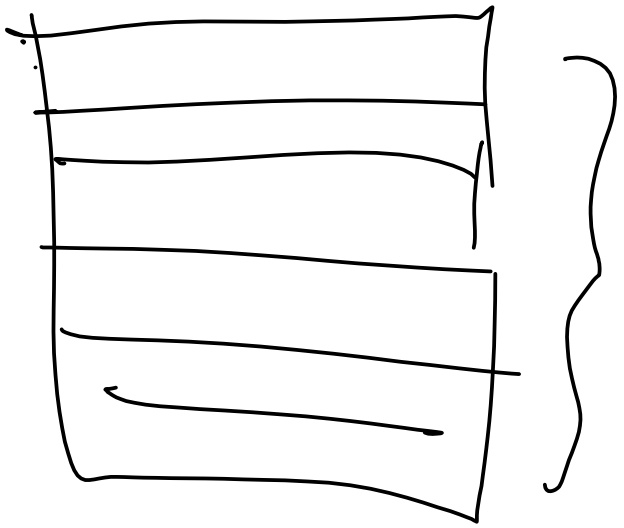
get: at that pos

found it → ✓

not found → go to next

empty → finished

Linear probing



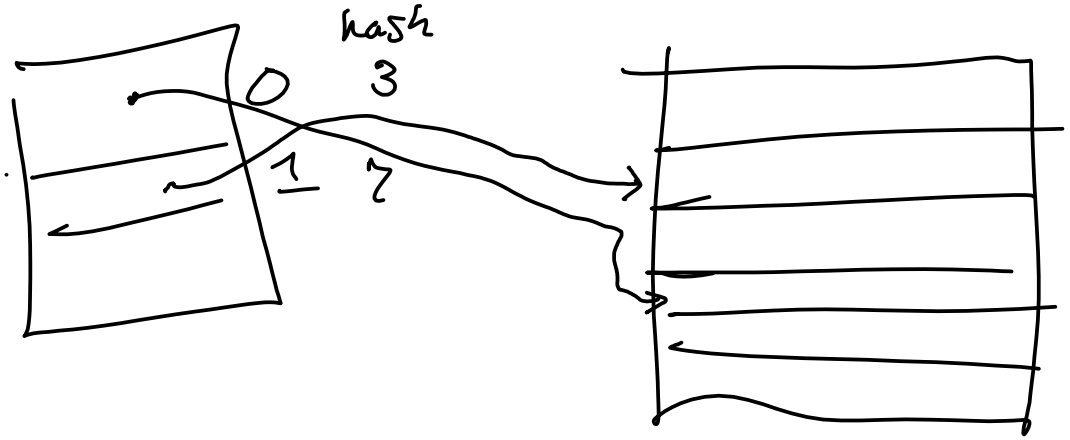
size = # mappings

buckets = "capacity"

n "buckets"

$$\text{load factor} = \frac{\text{size}}{\text{capacity}}$$

Maximum load factor



```
int hash(string s) {
    int acc = 0;
    for (int index = 0; index < s.length(); index++) {
        acc *= 31;
        acc += s[index];
    }
    return acc;
}
```