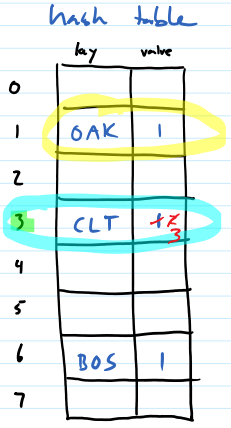
Map as Hash Table

```
20144987 4 4 4 CLT OAK X 17 1.00 X
20144756 2 4 4 CLT BOS X US 1.00 X
201441020 2 4 4 CLT MCO X US 1.00 X
201442094 3 4 4 CHS CLT - 16 1.00 X
20144214 3 4 4 IAH CLT - US 1.00 X
201441020 1 4 4 AVL CLT - 16 1.00 X
201441110 4 4 4 PHX OAK X US 1.00 X
20144794 3 4 4 CLE CLT - 16 1.00 X
201441578 4 5 4 BNA PHL - YX 1.00 X
201442030 4 4 4 DCA BHM X US 1.00 X
201441020 3 4 4 MCO CLT - US 1.00 X
```

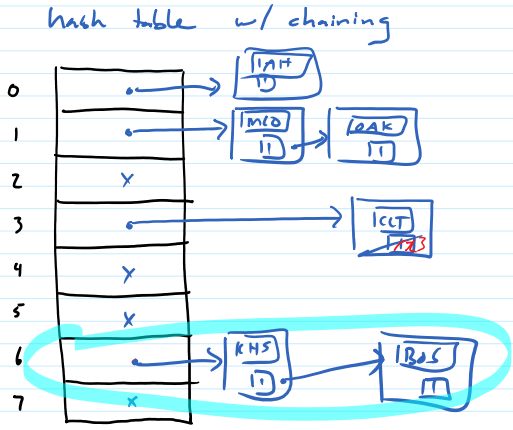


hash code — arbitrary but consistent

CLT 3
OAK 1
BOS 1
MCO 1
CHS
IAH
AVL

collision!

66827 % 8 = 3
70009 % 8 = 1
65958 % 8 = 6
76153 % 8 = 1




```
20144987 4 4 4 CLT OAK X 17 1.00 X
20144756 2 4 4 CLT BOS X US 1.00 X
201441020 2 4 4 CLT MCO X US 1.00 X
201442094 3 4 4 CHS CLT - 16 1.00 X
20144214 3 4 4 IAH CLT - US 1.00 X
201441020 1 4 4 AVL CLT - 16 1.00 X
201441110 4 4 4 PHX OAK X US 1.00 X
20144794 3 4 4 CLE CLT - 16 1.00 X
201441578 4 5 4 BNA PHL - YX 1.00 X
201442030 4 4 4 DCA BHM X US 1.00 X
201441020 3 4 4 MCO CLT - US 1.00 X
```

hash fxn

CLT 3
OAK 1
BOS 1
MCO 1
CHS 6
IAH
AVL

66827 % 8 = 3
70009 % 8 = 1
65958 % 8 = 6
76153 % 8 = 1
66762 % 8 = 6

contains_key, get, put:
1) compute hash(key)
2) compute remainder when dividing by size
3) sequential search of the list at that index

worst case O(n)

$$\text{load factor} = \frac{\#\text{ entries}}{\#\text{ slots}} = \alpha$$

if keys are evenly distributed, expected # keys/slot = α

expected running time for get/put/contains = expected keys/slot = α
(assuming O(1) for hash fxn)

```
20144987 4 4 4 CLT OAK X 17 1.00 X
20144756 2 4 4 CLT BOS X US 1.00 X
201441020 2 4 4 CLT MCO X US 1.00 X
201442094 3 4 4 CHS CLT - 16 1.00 X
20144214 3 4 4 IAH CLT - US 1.00 X
201441020 1 4 4 AVL CLT - 16 1.00 X
201441110 4 4 4 PHX OAK X US 1.00 X
20144794 3 4 4 CLE CLT - 16 1.00 X
201441578 4 5 4 BNA PHL - YX 1.00 X
201442030 4 4 4 DCA BHM X US 1.00 X
201441020 3 4 4 MCO CLT - US 1.00 X
```

hash table w/ open addressing
linear probing

remove (OAK)
get (AVL)

contains_key
put
get

1) compute hash(key)
2) compute remainder % # slots
3) sequential search through slots starting there, stopping when key found or at untouched slot

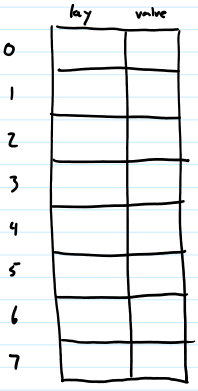
CLT 3 66827 % 8 = 3
OAK 1 70009 % 8 = 1
BOS 65958 % 8 = 6
MCO 76153 % 8 = 1
CHS 66762 % 8 = 6
IAH 77240 % 8 = 0
AVL 65207 % 8 = 7

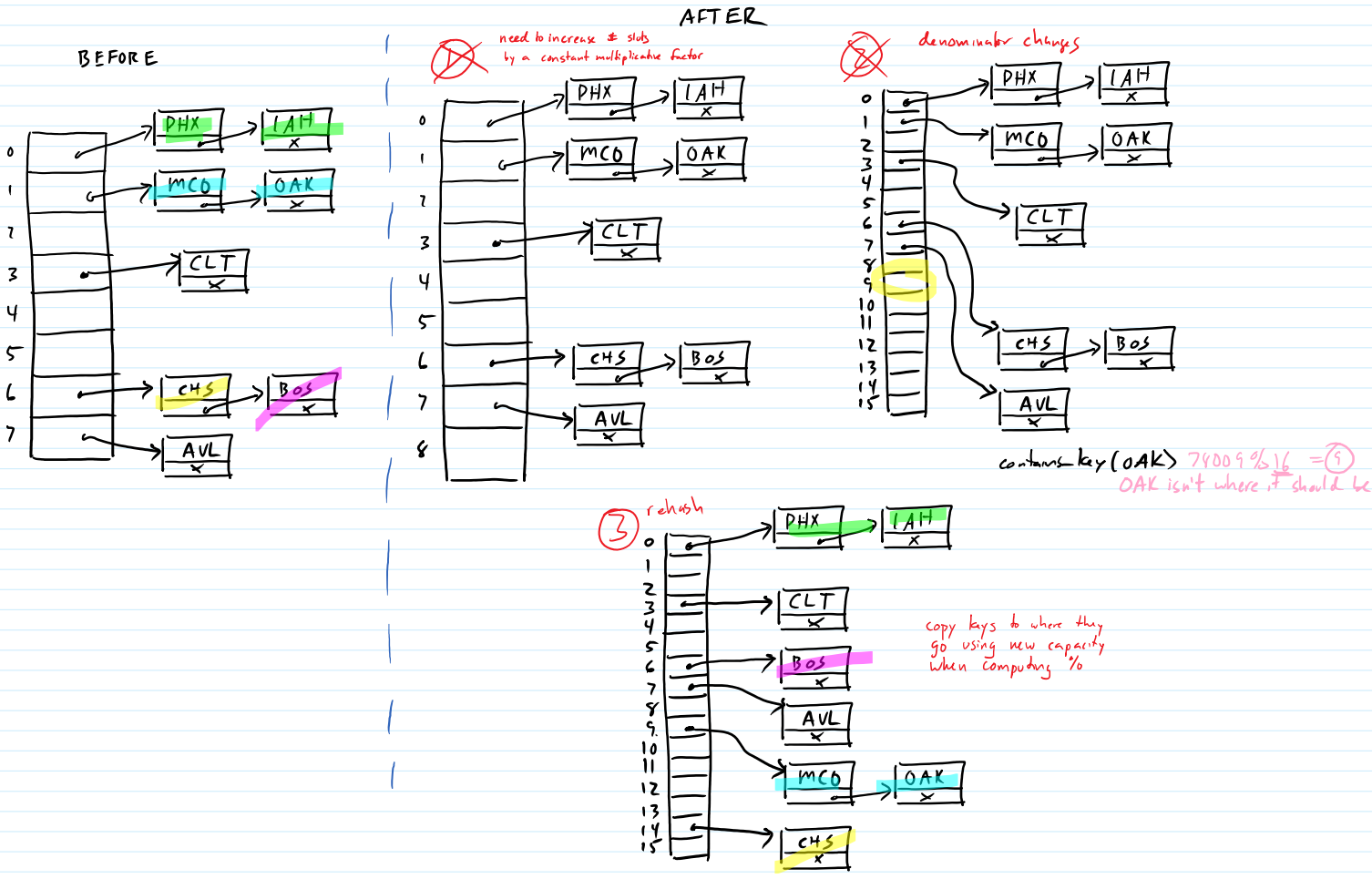
0	IAH	1
1		
2		
3	CLT	4
4		
5		
6	BOS	1
7	AVL	1
8		
9	OAK	1
10	MCO	1
11		
12		
13		
14	CHS	1
15		

expected time = expected time for sequential search (assuming O(1) for hash)
= expected size of cluster (consecutive used slots)
if $\alpha \leq \frac{1}{2}$, avg cluster size ≤ 2
so O(1) expected time

hash fxn

CLT
AVL
IAH
PHX
OAK




BEFORE
AFTER
need to increase # slots by a constant multiplicative factor
denominator changes
PHX
IAH
MCO
OAK
CLT
CHS
BOS
AVL
contains key(OAK) 74009 % 16 = 9
OAK isn't where it should be
rehash
copy keys to where they go using new capacity when computing %