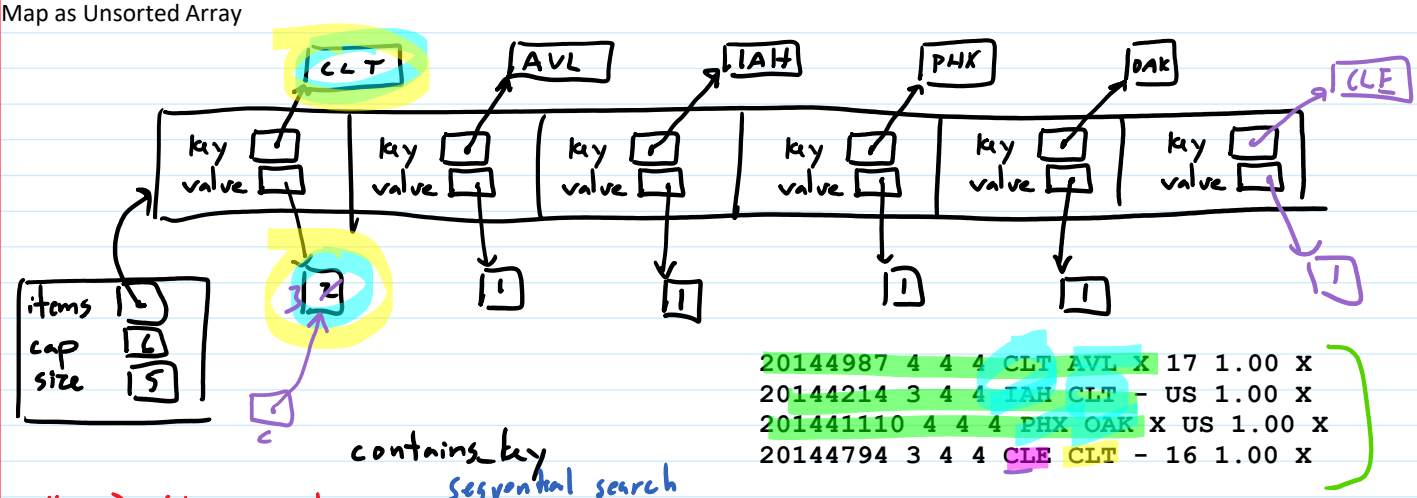
Map as Unsorted Array



containsKey
sequential search
O(n)

O(log n) binary search

get

O(log n)

sequential search
O(n)

O(n) binary search O(log n) insert O(n)

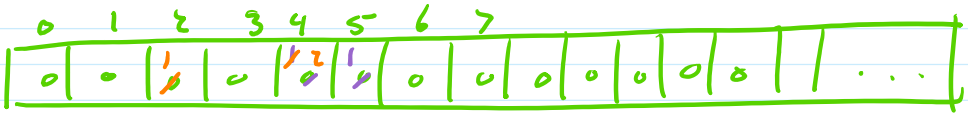
put - replaces existing value if key present
adds new (key, value) pair if key not present
] seq. search
O(n)

Sorted array

so $O(n^2)$ overall to count frequencies among n things

if we used integer codes instead of 3-4 letters

4	5
2	4
16	12
10	4



O(1) worst case per operation

O(n) to count frequencies among n things

Map as Hash Table

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

hash table

	key	value
0		
1		
2		
3	CLT	12
4		
5	IAH	1
6		
7	AVL	1

hash fxn = arbitrary but consistent value

	hash fxn	
CLT	3	
AVL	7	
IAH	21	21 % 8 = 5
PHX	805	805 % 8 = 5 collision
OAK		
CLE		
BOS		
MCO		

hash table w/ chaining
slots in hash table are lists
collisions resolved by adding to list

	key	value
0	BOS	1
1		
2		
3	CLT	12 → CLE 1
4	MCO	1
5	IAH	1 → PHX 1 → OAK 1
6		
7	AVL	1

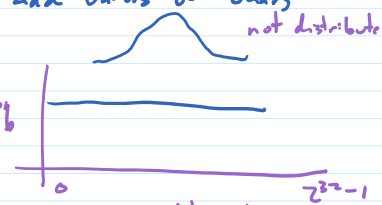
	hash fxn	
CLT	3	
AVL	7	
IAH	21	21 % 8 = 5
PHX	805	805 % 8 = 5 collision
OAK	477	477 % 8 = 5
CLE	51	51 % 8 = 3
BOS	0	
MCO	4	

contains_key, get, put { compute hash; compute remainder; seq. search list @ index

O(n) worst case (all things collide in same chain)

bad hash: take 1st char 256 not spread out

still bad: add values of chars not distributed evenly



good: freq

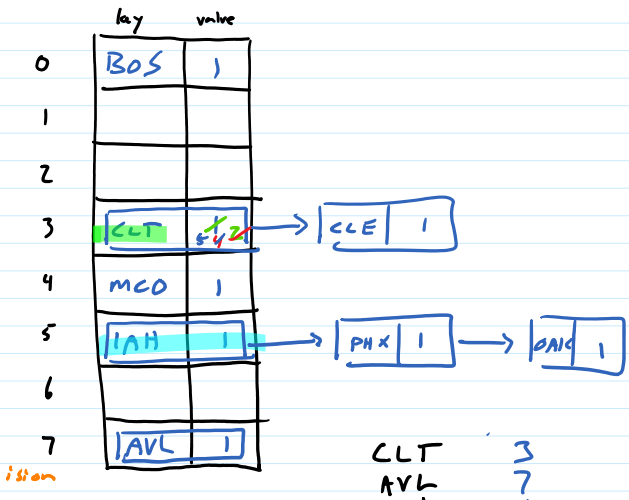
and hash similar keys to dissimilar values

before 9th new (key, value) resize hash table

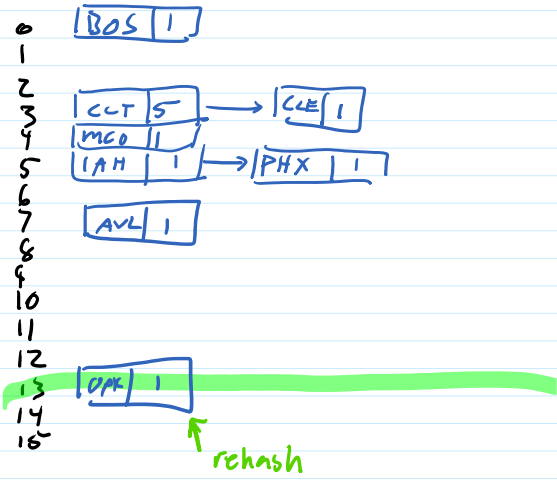
- keep load factor $\frac{\#keys}{\#slots} = \alpha \leq 1$

on avg, seq searches O(1)

still worst case O(n)

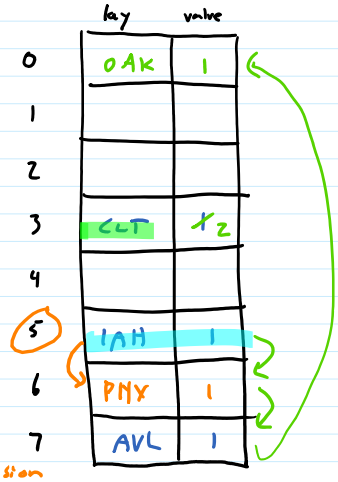


CLT	3
AVL	7
IAH	21
PHX	805
OAK	477
CLE	51
BOS	0
MCO	4



all operations still O(1) expected time
(if doubling size)

hash table w/ open addressing




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

	hash fxn
CLT	3
AVL	7
IAH	21
PHX	805
→ OAK	477
CLE	
BOS	
MCO	

21 % 8 = 5
805 % 8 = 5 collision

contains_key 1) compute hash
put 2) compute %
get 3) seq. search through slots until key found or empty or wrapped all the way around

O(n)

if $\alpha \leq \frac{1}{2}$ → O(1) expected time