CSE12 - Lecture 16

Wednesday, May 10, 2023 8:00 AM

PA5
PA2 Late-resubmit > due tonight

PA6 → due Tuesday

Hash Function (same as previous)

```
int getIndex(String k) {
    return k.length;
}
```

# of buckets - 4
(i.e. the size of the array)

expandCapacity() called in set()

LoadFactor – 0.75 → # elements / capacity → size / array length

```
set("Smith", 1);
set("Johnson", 2);
set("Williams", 3);
set("Brown", 4);
set("Jones", 5);
set("Garcia", 6);
set("Miller", 7);
set("Davis", 8);
set("Rodriguez", 9);
set("Martinez", 10);
```

Draw the picture of the HashTable using Separate Chaining (using expandCapacity)

Does the run-time change with expandCapacity()?

Yes

What is the run-time for this HashTable (do picture first):

set()

Worst Case $\Theta(N^2)$

Best Case: $\Theta(1)$ ← this is much more likely to happen

What conditions make up the best case for set()?

no expand capacity, empty or nearly empty

get()

Worst Case $\Theta(N)$

Best Case: $\Theta(1)$ ← more likely to happen

What conditions make up the best case for get()?

even distribution, empty or nearly empty

Why is the hash function important?

even distribution
→ less collisions

set

$(\Theta(1) + \overset{\text{ALEC}}{\Theta(N)}) * N \rightarrow \Theta(N) + \Theta(N^2)$

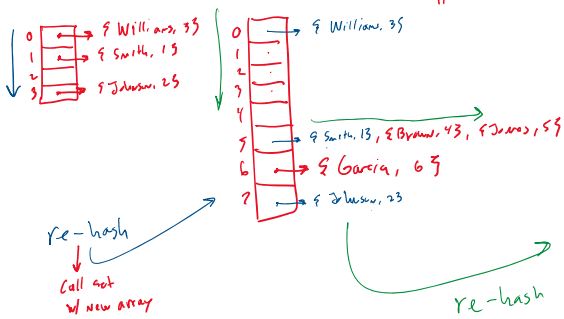
Name: _____ PID: _____ Code: 1 9 10

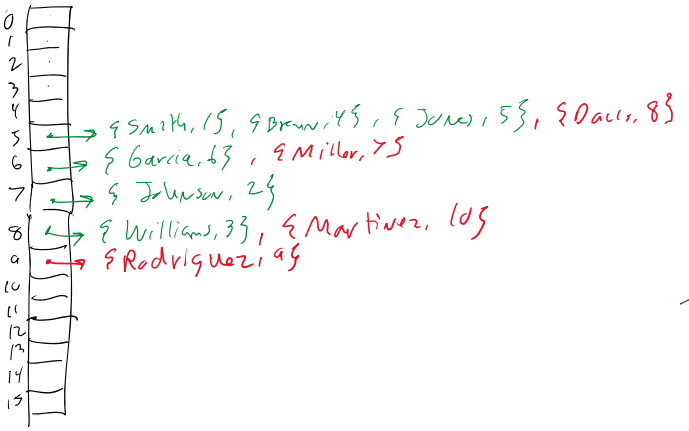
hash % capacity

LoadFactor – 0.75

```
set("Smith", 1);
set("Johnson", 2);
set("Williams", 3);
set("Brown", 4);
set("Jones", 5);
set("Garcia", 6);
set("Miller", 7);
set("Davis", 8);
set("Rodriguez", 9);
set("Martinez", 10);
```

hash	index	LF
5	~~5~~	0
7	~~7~~	1/4 .25
8	0 or 8	2/4 .5
5	5	3/4 ~~.75~~ 3/8
5	5	4/8 .5
6	6	5/8 ~~.625~~ 6/16
6	6	6/8
5	5	7/16
9	9	8/16
8	8	9/16





clustering
↓
bad hash function