CSE12 - Lecture 16 - B00

Sunday, October 30, 2022 9:00 AM

PA 5 due tomorrow
PA 2 Late / Resubmit due tomorrow

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

```
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()?

What is the run-time for this HashTable (do picture first): N * 2N

set()
Worst Case $\Theta(N^2)$
Best Case: $\Theta(1)$

What conditions make up the best case for set()?
no expand capacity, empty or many empty buckets

get()
Worst Case $\Theta(N)$
Best Case: $\Theta(1)$

What conditions make up the best case for get()?
even distribution, 0/1, ~2 elements per bucket

Why is the hash function important?
even distribution
less collisions

LoadFactor – 0.75 → size / capacity >= → expand capacity

	hash	index	lf
set("Smith", 1);	5	5	1/4 → 0
set("Johnson", 2);	7	7	2/4 → .25
set("Williams", 3);	8	0	3/4 → .5
set("Brown", 4);	5	1	3/4 → .75 . 3/8
set("Jones", 5);	5	2	4/8 → .5
set("Garcia", 6);	6	3	5/8
set("Miller", 7);	6	4	6/8 → .75 , 6/16
set("Davis", 8);	3	5	7/16
set("Rodriguez", 9);	9	6	8/16
set("Martinez", 10);	8	7	9/16